ACTIONSCRIPT® 3.0
Entwicklerhandbuch

Rechtliche Hinweise

Rechtliche Hinweise finden Sie unter http://help.adobe.com/de_DE/legalnotices/index.html.

Letzte Aktualisierung 12.12.2014

Inhalt

Kapitel 1: Arbeiten mit Datum und Zeit
Verwalten von Kalenderdatum und -zeit 1
Steuern von Zeitintervallen 4
Beispiel für Datum und Uhrzeit: Einfache Analoguhr 6

Kapitel 2: Arbeiten mit Strings
Grundlagen von Strings 10
Erstellen von Strings 11
Length-Eigenschaft 12
Arbeiten mit Zeichen in Strings 12
Vergleichen von Strings 13
Stringdarstellung anderer Objekte 14
Verketten von Strings 14
Suchen von Teilstrings und Mustern in Strings 15
Umwandeln der Groß- und Kleinschreibung von Strings 19
Strings-Beispiel: ASCII-Grafik 20

Kapitel 3: Verwenden von Arrays
Grundlagen von Arrays 25
Indizierte Arrays 27
Assoziative Arrays 39
Mehrdimensionale Arrays 43
Klonen von Arrays 44
Erweitern der Array-Klasse 45
Arrays-Beispiel: PlayList 50

Kapitel 4: Verarbeiten von Fehlern
Grundlagen der Fehlerverarbeitung 55
Fehlertypen 56
Fehlerverarbeitung in ActionScript 3.0 59
Arbeiten mit den Debugger-Versionen der Flash-Laufzeitumgebungen 61
Verarbeiten synchroner Fehler in einer Anwendung 62
Erstellen benutzerdefinierter Fehlerklassen 66
Reagieren auf Fehlerereignisse und Status 67
Vergleich der Error-Klassen 71
Beispiel für die Fehlerverarbeitung: CustomErrors-Anwendung 74

Kapitel 5: Verwenden von regulären Ausdrücken
Grundlagen von regulären Ausdrücken 81
Syntax regulärer Ausdrücke 82
Methoden für das Verwenden regulärer Ausdrücke bei Strings 96
Beispiel für reguläre Ausdrücke: Wiki-Parser 97

Kapitel 6: XML-Verarbeitung
Grundlagen von XML ... 102
E4X-Ansatz der XML-Verarbeitung ... 105
XML-Objekte ... 107
XMLList-Objekte ... 109
Initialisieren von XML-Variablen ... 110
Zusammenstellen und Transformieren von XML-Objekten ... 112
Durchsuchen von XML-Strukturen ... 113
Verwenden von XML-Namespaces ... 118
XML-Typumwandlung ... 118
Lesen externer XML-Dokumente ... 120
Beispiel für XML in ActionScript: Laden von RSS-Daten aus dem Internet ... 121

Kapitel 7: Verwenden der nativen JSON-Funktionalität
Überblick über die JSON-API ... 125
Definieren von JSON-Verhalten ... 126

Kapitel 8: Verarbeiten von Ereignissen
Grundlagen der Ereignisverarbeitung ... 133
Unterschiede der Ereignisverarbeitung in ActionScript 3.0 im Vergleich mit früheren Versionen ... 135
Der Ereignisablauf ... 137
Ereignisobjekte ... 139
Ereignis-Listener ... 143
Beispiel für die Ereignisverarbeitung: Alarm Clock ... 150

Kapitel 9: Verwenden von Anwendungsdomänen

Kapitel 10: Programmieren von Anzeigeobjekten
Grundlagen der Programmierung von Anzeigeobjekten ... 162
Wichtige Anzeigeklassen ... 165
Vorteile von Anzeigelisten ... 167
Verwenden von Anzeigeobjekten ... 169
Bearbeiten von Anzeigeobjekten ... 185
Animieren von Objekten ... 205
Bühnenausrichtung ... 208
Dynamisches Laden von Anzeigeinhalten ... 211
Beispiel für ein Anzeigeobjekt: SpriteArranger ... 217

Kapitel 11: Verwenden von geometrischen Objekten
Grundlagen von geometrischen Objekten ... 224
Verwenden von Point-Objekten ... 225
Verwenden von Rectangle-Objekten ... 227
Verwenden von Matrix-Objekten ... 230
Geometrie-Beispiel: Anwenden einer Matrixtransformation auf ein Anzeigeobjekt ... 232

Kapitel 12: Verwenden der Zeichnungs-API
Grundlagen der Zeichnungs-API ... 236
Die Graphics-Klasse ... 237

Zeichnen von Linien und Kurven 238
Zeichnen von Formen mit integrierten Methoden 240
Erstellen von Farbverlaufslinien und -füllungen 241
Verwenden der Math-Klasse mit Zeichnungsmethoden 245
Animation mit der Zeichnungs-API 246
Beispiel für die Zeichnungs-API: Algorithmic Visual Generator 247
Erweiterte Einsatzmöglichkeiten der Zeichnungs-API 250

Kapitel 13: Verwenden von Bitmaps
Grundlagen zum Arbeiten mit Bitmaps 258
Bitmap-Klasse und BitmapData-Klasse 260
Bearbeiten von Pixelwerten 262
Kopieren von Bitmapdaten 265
Komprimieren von Bitmapdaten 266
Erstellen von Texturen mit Störungsfunktionen 267
Durchführen eines Bildlaufs in Bitmaps 269
Nutzen von MIP-Mapping 270
Bitmap-Beispiel: Animated Spinning Moon 271
Asynchrone Dekodierung von Bitmapbildern 282

Kapitel 14: Anwenden von Filtern auf Anzeigeobjekte
Grundlagen der Anwendung von Filtern auf Anzeigeobjekte 286
Erstellen und Anwenden von Filtern 287
Verfügbare Anzeigefilter 294
Beispiel für das Filtern von Anzeigeobjekten: Filter Workbench 312

Kapitel 15: Arbeiten mit Pixel Bender-Shadern
Grundlagen von Pixel Bender-Shadern 321
Laden oder Einbetten eines Shaders 323
Zugreifen auf Shader-Metadaten 325
Angeben von Shader-Eingaben und -Parameterwerten 326
Verwenden eines Shaders 331

Kapitel 16: Verwenden von Movieclips
Grundlagen zu Movieclips 344
Verwenden von MovieClip-Objekten 345
Steuern der Wiedergabe von Movieclips 345
Erstellen von MovieClip-Objekten mit ActionScript 348
Laden einer externen SWF-Datei 351
Movieclip-Beispiel: RuntimeAssetsExplorer 353

Kapitel 17: Arbeiten mit Bewegungs-Tweens
Grundlagen von Bewegungs-Tweens 358
Kopieren von Bewegungs-Tween-Skripts in Flash 359
Einbinden von Bewegungs-Tween-Skripts 360
Beschreiben der Animation 361
Hinzufügen von Filtern 364
Verknüpfen von Bewegungs-Tweens mit den zugehörigen Anzeigeobjekten 366

Kapitel 18: Arbeiten mit inverser Kinematik

Grundlagen der inversen Kinematik 368
Animieren von IK-Skeletten – Überblick 369
Abrufen von Informationen über IK-Skelette 371
Instanziieren eines IKMover-Objekts und Einschränken seiner Bewegung 371
Bewegen eines IK-Skeletts 372
Verwenden von Federn 373
Verwenden von IK-Ereignissen 374

Kapitel 19: Arbeiten mit drei Dimensionen (3D)

Grundlagen von 3D-Anzeigeobjekten 375
Informationen zu 3D-Anzeigeobjekten in Flash Player und der AIR-Laufzeitumgebung 376
Erstellen und Verschieben dreidimensionaler Anzeigeobjekte 378
Projizieren von dreidimensionalen Objekten auf eine Fläche 380
Beispiel: Perspektivische Projektion 382
Durchführen komplexer 3D-Transformationen 385
Verwenden von Dreiecken für 3D-Effekte 388

Kapitel 20: Grundlagen der Textverarbeitung

Kapitel 21: Verwenden der TextField-Klasse

Anzeigen von Text 399
Auswählen und Bearbeiten von Text 403
Erfassen von Texteingaben 405
Einschränken der Texteingabe 406
Formatieren von Text 407
Erweiterte Textdarstellung 411
Verwenden von statischem Text 414
TextField-Beispiel: Textformatierung im Zeitungsstil 415

Kapitel 22: Verwenden der Flash Text Engine

Erstellen und Anzeigen von Text 425
Verarbeiten von Ereignissen in FTE 430
Formatieren von Text 433
Arbeiten mit Schriftarten 438
Steuern von Text 440
Flash Text Engine-Beispiel: Zeichnungslayout 446

Kapitel 23: Verwenden des Text Layout Framework

Übersicht über das Text Layout Framework 455
Verwenden des Text Layout Framework 457
Strukturieren von Text mit dem TLF 462
Formatieren von Text mit dem TLF 466
Importieren und Exportieren mit TLF 467
Verwalten von Textcontainern mit dem TLF 468
Aktivieren von Textauswahl, Bearbeitung und Rückgängigmachen mit dem TLF 469
Ereignisverarbeitung mit dem TLF 469
Positionieren von Bildern innerhalb von Text 470

Kapitel 24: Arbeiten mit Sounds

Grundlagen zum Arbeiten mit Sound 471
Soundarchitektur 472
Laden von externen Sounddateien 474
Verwenden eingebetteter Sounds 476
Verwenden von Streaming-Sounddateien 478
Verwenden von dynamisch generierten Audiodaten 479
Wiedergeben von Sounds 481
Sicherheitsüberlegungen beim Laden und Wiedergeben von Sounds 485
Steuern der Lautstärke und Richtungseinstellung des Sounds 486
Verwenden von Sound-Metadaten 488
Zugreifen auf Raw-Sounddaten 488
Erfassen von Soundeingaben 492
Sound-Beispiel: Podcast-Player 498

Kapitel 25: Verwenden von Videos

Grundlagen zu Video 505
Videoformate 506
Video-Klasse 510
Laden von Videodateien 510
Steuern der Videowiedergabe 511
Videowiedergabe im Vollbildmodus 513
Streamen von Videodateien 517
Cue-Points 517
Schreiben von Rückrufmethoden für Metadaten und Cue-Points 518
Verwenden von Cue-Points und Metadaten 524
Überwachen der NetStream-Aktivität 534
Erweiterte Themen für Videodateien 537
Video-Beispiel: Video Jukebox 539
Verwenden der StageVideo-Klasse für die hardwarebeschleunigte Darstellung 545

Kapitel 26: Arbeiten mit Kameras

Camera-Klasse 554
Anzeigen des Kamerainhalts auf dem Bildschirm 555
Entwerfen der Kamera-Anwendung 555
Herstellen einer Verbindung mit der Kamera eines Benutzers 556
Prüfen auf installierte Kameras 556
Erfassen der Zugriffsberechtigungen für eine Kamera 557
Maximieren der Kamera-Videoqualität 559
Überwachen des Kamerastatus 560

Kapitel 27: Digitale Rechteverwaltung

Arbeitsablauf für geschützte Inhalte 563
DRM-relevante Mitglieder und Ereignisse der NetStream-Klasse 570
Verwenden der DRMStatusEvent-Klasse 572
Verwenden der DRMAuthenticateEvent-Klasse 573
Verwenden der DRMErrorEvent-Klasse 577

Verwenden der DRMManager-Klasse 577
Verwenden der DRMContentData-Klasse 579
Aktualisieren von Flash Player zur Unterstützung von Adobe Access 579
Out-of-Band-Lizenzen 581
Domänenunterstützung 582
Abspielen verschlüsselter Inhalte mit Domänenunterstützung 583
Lizenzvorschau 583
Bereitstellen von Inhalten 584
Open Source Media Framework 584

Kapitel 28: Hinzufügen von PDF-Inhalten in AIR
Erkennen der PDF-Funktionalität 587
Laden von PDF-Inhalten 588
Skripterstellung für PDF-Inhalte 589
Bekannte Beschränkungen für PDF-Inhalt in AIR 590

Kapitel 29: Grundlagen der Benutzerinteraktion
Erfassen von Benutzereingaben 592
Fokusverwaltung 593
Erkennen von Eingabetypen 594

Kapitel 30: Tastatureingabe
Erfassen von Tastatureingaben 596
Verwenden der IME-Klasse 598
Virtuelle Tastaturen 603

Kapitel 31: Mauseingabe
Erfassen von Mauseingaben 612
Beispiel für Mauseingabe: WordSearch 615

Kapitel 32: Eingabe per Berührung, Multitouch und Gesten
Grundlagen der Berührungseingabe 620
Erkennung der Berührungsunterstützung 622
Verarbeitung von Berührungsereignissen 623
Berühren und Ziehen 627
Verarbeitung von Gestenereignissen 628
Fehlerbehebung 632

Kapitel 33: Kopieren und Einfügen
Grundlagen zum Kopieren und Einfügen 635
Lesen aus der und Schreiben in die System-Zwischenablage 636
Kopieren und Einfügen von HTML in AIR 636
Datenformate in der Zwischenablage 638

Kapitel 34: Accelerometer-Eingabe
Überprüfen der Accelerometer-Unterstützung 645
Erkennen von Accelerometer-Änderungen 646

Kapitel 35: Ziehen und Ablegen in AIR

Grundlagen zur Drag & Drop-Funktion in AIR 648
Unterstützung des Herausziehens 650
Unterstützung des Hineinziehens 653
Drag & Drop in HTML 656
Herausziehen von Daten aus einem HTML-Element 660
Hineinziehen von Daten in ein HTML-Element 661
Beispiel: Überschreiben des Standardverhaltens für das Hineinziehen von Elementen in HTML-Elemente 661
Abgelegte Dateien in anwendungsfremden HTML-Sandboxen 663
Ablegen von Dateizusagen 665

Kapitel 36: Arbeiten mit Menüs

Grundlagen von Menüs 673
Erstellen nativer Menüs (AIR) 680
Kontextmenüs in HTML (AIR) 682
Anzeigen von nativen Popupmenüs (AIR) 683
Umgang mit Menüereignissen 684
Beispiel für natives Menü: Fenster- und Anwendungsmenü (AIR) 685

Kapitel 37: Taskleisten-Symbol in AIR

Taskleisten-Symbole 689
Dock-Symbole 690
Infobereich-Symbole 690
Taskleisten-Symbole und -Schaltflächen für Fenster 692

Kapitel 38: Arbeiten mit dem Dateisystem

Verwenden der FileReference-Klasse 695
Verwenden der AIR-Dateisystem-API 710

Kapitel 39: Speichern lokaler Daten

Gemeinsame Objekte 747
Verschlüsselter lokaler Speicher 757

Kapitel 40: Arbeiten mit lokalen SQL-Datenbanken in AIR

Lokale SQL-Datenbanken 761
Erstellen und Modifizieren von Datenbanken 766
Bearbeiten von SQL-Datenbankdaten 773
Verwenden von synchronen und asynchronen Datenbankoperationen 802
Verwenden von Verschlüsselung mit SQL-Datenbanken 807
Strategien für die Arbeit mit SQL-Datenbanken 825

Kapitel 41: Arbeiten mit Byte-Arrays

Lesen und Schreiben von ByteArrays 828
ByteArray-Beispiel: Lesen von .zip-Dateien 834

Kapitel 42: Grundlagen zu Netzwerken und Kommunikation

Netzwerkschnittstellen 842
Änderungen an Netzwerkverbindungen 844
DNS-Datensätze 846

Kapitel 43: Sockets
TCP-Sockets 849
UDP-Sockets (AIR) 861
IPv6-Adressen 863

Kapitel 44: HTTP-Kommunikation
Laden von externen Daten 865
Webdienstanforderungen 874
Öffnen einer URL in einer anderen Anwendung 882

Kapitel 45: Kommunikation mit anderen Flash Player- und AIR-Instanzen
LocalConnection-Klasse 884
Senden von Meldungen zwischen zwei Anwendungen 886
Herstellen von Verbindungen mit Inhalten in verschiedenen Domänen und mit AIR-Anwendungen 889

Kapitel 46: Kommunikation mit nativen Prozessen in AIR
Überblick über die Kommunikation mit nativen Prozessen 891
Starten und Schließen eines nativen Prozesses 892
Kommunikation mit einem nativen Prozess 893
Sicherheitsaspekte bei der Kommunikation mit nativen Prozessen 895

Kapitel 47: Verwenden der externen API
Grundlagen der Verwendung der externen API 896
Anforderungen und Vorteile externer APIs 899
Verwenden der ExternalInterface-Klasse 900
Beispiel für externe API: Kommunikation zwischen ActionScript und JavaScript in einem Webbrowser 904

Kapitel 48: Validierung von XML-Signaturen in AIR
Grundlagen zur Validierung von XML-Signaturen 911
Über XML-Signaturen 916
Implementieren der IURIDereferencer-Schnittstelle 919

Kapitel 49: Clientsystem-Umgebung
Grundlagen der Clientsystem-Umgebung 927
Verwenden der System-Klasse 928
Verwenden der Capabilities-Klasse 929
Capabilities-Beispiel: Ermitteln von Systemeigenschaften 930

Kapitel 50: Aufrufen und Beenden von AIR-Anwendungen
Aufrufen der Anwendung 935
Erfassen von Befehlszeilenargumenten 937
Aufrufen einer AIR-Anwendung bei der Benutzeranmeldung 940
Aufrufen von AIR-Anwendungen aus dem Browser 941
Schließen der Anwendung 943

Kapitel 51: Arbeiten mit AIR-Laufzeit- und Betriebssysteminformationen
Verwalten von Dateiverknüpfungen 945
Abrufen von Laufzeitversion und Patchebene 946

Erkennen von AIR-Funktionalität 946
Nachverfolgen der Benutzerpräsenz 947

Kapitel 52: Arbeiten mit nativen AIR-Fenstern
Grundlagen von nativen Fenstern in AIR 948
Erstellen von Fenstern 956
Verwalten von Fenstern 965
Warten auf Window-Ereignisse 976
Anzeigen von Fenstern im Vollbildmodus 977

Kapitel 53: Anzeigebildschirme in AIR
Grundlagen zu Anzeigebildschirmen in AIR 980
Aufzählen von Bildschirmen 981

Kapitel 54: Drucken
Grundlagen zum Drucken 985
Drucken einer Seite 986
Aufgaben in der Flash-Laufzeitumgebung und Drucken im System 987
Einrichten von Größe, Skalierung und Ausrichtung 990
Erweiterte Drucktechniken 992
Druckbeispiel: Mehrseitiger Druck 994
Druckbeispiel: Skalieren, Zuschneiden und Anpassen 996
Druckbeispiel: Seiteneinrichtung und Druckoptionen 998

Kapitel 55: Geolokation
Erkennen von Geolokationsänderungen 1001

Kapitel 56: Internationalisierung von Anwendungen
Internationalisierung von Anwendungen – Grundlagen 1005
Übersicht über das flash.globalization-Paket 1006
Bestimmen des Gebietsschemas 1008
Formatieren von Zahlen 1009
Formatieren von Währungswerten 1012
Formatieren von Datum und Uhrzeit 1014
Sortieren und Vergleichen von Strings 1016
Umwandlung der Groß- und Kleinschreibung 1017
Beispiel: Internationalisierung einer Börsenticker-Anwendung 1018

Kapitel 57: Lokalisieren von Anwendungen
Auswählen eines Gebietsschemas 1023
Lokalisieren von Flex-Inhalten 1024
Lokalisieren von Flash-Inhalten 1024
Lokalisieren von AIR-Anwendungen 1025
Lokalisieren von Datum, Uhrzeit und Währungen 1025

Kapitel 58: Einführung in die HTML-Umgebung
Überblick über die HTML-Umgebung 1027
AIR und WebKit 1030

Kapitel 59: Programmieren mit HTML und JavaScript in AIR

HTMLLoader-Klasse 1047
Vermeiden von sicherheitsbezogenen JavaScript-Fehlern 1049
Zugreifen auf AIR API-Klassen aus JavaScript 1054
URLs in AIR 1056
Bereitstellen von ActionScript-Objekten für JavaScript 1057
Zugreifen auf HTML-DOM- und JavaScript-Objekte mit ActionScript 1058
Einbetten von SWF-Inhalten in HTML 1060
Verwenden von ActionScript-Bibliotheken in HTML-Seiten 1063
Konvertieren von Date- und RegExp-Objekten 1065
Manipulieren eines HTML-Stylesheets mit ActionScript 1065
Cross-Scripting von Inhalten in unterschiedlichen Sicherheits-Sandboxen 1067

Kapitel 60: Skripterstellung von AIR-HTML-Containern

Anzeigeeigenschaften des HTMLLoader-Objekts 1072
Bildlauf von HTML-Inhalt 1075
Zugreifen auf die HTML-Verlaufsliste 1076
Festlegen des Benutzer-Agent-Strings, der beim Laden von HTML-Inhalt verwendet wird 1077
Festlegen der mit HTML-Inhalt verwendeten Zeichenkodierung 1077
Definieren browserähnlicher Benutzeroberflächen für HTML-Inhalt 1078
Erstellen von Unterklassen für die HTMLLoader-Klasse 1089

Kapitel 61: Verarbeiten HTML-bezogener Ereignisse in AIR

HTMLLoader-Ereignisse 1091
Verarbeiten von DOM-Ereignissen mit ActionScript 1092
Antworten auf nicht erfasste JavaScript-Ausnahmen 1092
Verarbeiten von Laufzeitereignissen mit JavaScript 1094

Kapitel 62: Anzeigen von HTML-Inhalt in mobilen Anwendungen

StageWebView-Objekte 1098
Inhalt 1099
Navigationsereignisse 1100
Verlauf 1101
Fokus 1102
Bitmaperfassung 1104

Kapitel 63: Verwenden von Workern für Parallelität

Informationen zu Workern und Parallelität 1106
Erstellen und Verwalten von Workern 1107
Kommunikation zwischen Workern 1110

Kapitel 64: Sicherheit

Überblick über die Sicherheit der Flash-Plattform 1115
Sicherheits-Sandboxen 1117
Steuerung der Berechtigungen 1121
Einschränken von Netzwerk-APIs 1130
Sicherheit im Vollbildmodus 1133
Sicherheit im interaktiven Vollbildmodus 1134

Laden von Inhalten 1135
Cross-Scripting 1138
Zugriff auf geladene Medien als Daten 1142
Laden von Daten 1144
Laden von eingebetteten Inhalten aus SWF-Dateien, die in eine Sicherheitsdomäne importiert wurden 1148
Arbeiten mit Legacy-Inhalten 1148
Einstellen der LocalConnection-Berechtigungen 1149
Steuern des externen URL-Zugriffs 1149
Gemeinsame Objekte 1151
Zugriff auf Kamera, Mikrofon, Zwischenablage, Maus und Tastatur 1153
AIR-Sicherheit 1153

Kapitel 65: Verwendung der ActionScript-Beispiele
Beispielarten 1176
Ausführen von ActionScript 3.0-Beispielen in Flash Professional 1178
Ausführen von ActionScript 3.0-Beispielen in Flash Builder 1179
Ausführen von ActionScript 3.0-Beispielen auf Mobilgeräten 1181

Kapitel 66: SQL-Unterstützung in lokalen Datenbanken
Unterstützte SQL-Syntax 1186
Unterstützte Datentypen 1209

Kapitel 67: SQL-Fehlerdetailmeldungen, IDs und Argumente

Kapitel 68: Adobe Graphics Assembly Language (AGAL)
AGAL-Bytecodeformat 1220

Kapitel 1: Arbeiten mit Datum und Zeit

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die zeitliche Abstimmung ist nicht alles, sie spielt jedoch bei Softwareanwendungen in der Regel eine wesentliche Rolle. ActionScript 3.0 enthält leistungsstarke Funktionen zum Verwalten von Kalenderdaten, Uhrzeiten und Zeitintervallen. Zwei Hauptklassen enthalten die wichtigsten Funktionen für die zeitliche Abstimmung: die Date-Klasse und die neue Timer-Klasse im flash.utils-Paket.

Datums- und Uhrzeitangaben sind in ActionScript-Programmen häufig vorkommende Informationstypen. Beispielsweise kann es u. a. erforderlich sein, den aktuellen Wochentag zu ermitteln oder festzustellen, wie viel Zeit ein Benutzer in einem bestimmten Bildschirm verbringt. In ActionScript können Sie die Date-Klasse verwenden, um einen bestimmten Zeitpunkt (einschließlich Datums- und Uhrzeitangaben) anzugeben. Eine Date-Instanz enthält Werte für die einzelnen Datums- und Uhrzeiteinheiten wie Jahr, Monat, Tag, Wochentag, Stunden, Minuten, Sekunden, Millisekunden und Zeitzone. Für anspruchsvollere Einsatzzwecke umfasst ActionScript auch die Timer-Klasse, mit der Sie Aktionen nach einer bestimmten Verzögerung oder in festgelegten Zeitabständen ausführen können.

Verwandte Hilfethemen

Date

flash.utils.Timer

Verwalten von Kalenderdatum und -zeit

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Alle Funktionen zum Verwalten von Kalenderdaten und Uhrzeiten in ActionScript 3.0 sind in der Date-Hauptklasse konzentriert. Die Date-Klasse enthält Methoden und Eigenschaften, mit denen Sie Datums- und Uhrzeitangaben entweder in der Standardweltzeit (UTC, Coordinated Universal Time) oder in der spezifischen Ortszeit einer Zeitzone verwalten können. Bei der UTC handelt es sich um eine Standardzeitdefinition, die der Greenwich Mean Time (GMT) entspricht.

Erstellen von Date-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Date-Klasse ist mit Abstand eine der vielseitigsten Konstruktormethoden aller Hauptklassen. Sie können diese Klasse auf vier verschiedene Arten aufrufen.

Wenn erstens keine Parameter angegeben werden, gibt der `Date()`-Konstruktor ein Date-Objekt mit den aktuellen Datums- und Uhrzeitangaben in der Ortszeit der entsprechenden Zeitzone zurück. Beispiel:

```
var now:Date = new Date();
```

Wenn zweitens ein einzelner numerischer Parameter angegeben wird, wird dieser im `Date()`-Konstruktor als Anzahl der Millisekunden seit dem 1. Januar 1970 interpretiert und ein entsprechendes Date-Objekt wird zurückgegeben. Beachten Sie, dass der übergebene Wert als Millisekunden seit dem 1. Januar 1970 (UTC) interpretiert wird. Beim Date-Objekt werden jedoch Werte in der entsprechenden Ortszeit angegeben, es sei denn, Sie verwenden UTC-spezifische Methoden, um die Werte abzurufen und anzuzeigen. Wenn Sie ein neues Date-Objekt mit einem einzelnen milliseconds-Parameter erstellen, beachten Sie daher dabei den Zeitunterschied zwischen der Ortszeit und der Weltzeit. Mit den folgenden Anweisungen wird ein Date-Objekt erstellt, das auf Mitternacht des 1. Januar 1970 (UTC) gesetzt ist:

```
var millisecondsPerDay:int = 1000 * 60 * 60 * 24;
// gets a Date one day after the start date of 1/1/1970
var startTime:Date = new Date(millisecondsPerDay);
```

Sie können drittens mehrere numerische Parameter an den `Date()`-Konstruktor übergeben. Diese Parameter werden jeweils als Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde verarbeitet. Ein entsprechendes Date-Objekt wird zurückgegeben. Bei diesen Eingabeparametern wird davon ausgegangen, dass sie in Ortszeit und nicht gemäß der Weltzeit (UTC) angegeben werden. Mit den folgenden Anweisungen wird ein Date-Objekt erstellt, das auf Mitternacht des 1. Januar 2000, Ortszeit gesetzt ist:

```
var millenium:Date = new Date(2000, 0, 1, 0, 0, 0, 0);
```

Sie können viertens einen einzelnen Stringparameter an den `Date()`-Konstruktor übergeben. Dieser String wird in Datums- oder Zeitkomponenten geparst. Anschließend wird ein entsprechendes Date-Objekt zurückgegeben. Bei dieser Vorgehensweise empfiehlt es sich, den `Date()`-Konstruktor in einen `try..catch`-Codeblock einzuschließen, um eventuelle Parsingfehler abzufangen. Der `Date()`-Konstruktor akzeptiert verschiedene Stringformate (eine Liste dieser Formate finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform). Mit der folgenden Anweisung wird ein neues Date-Objekt mit einem Stringwert initialisiert:

```
var nextDay:Date = new Date("Mon May 1 2006 11:30:00 AM");
```

Wenn der `Date()`-Konstruktor den Stringparameter nicht erfolgreich analysieren kann, wird eine Ausnahme ausgelöst. Das resultierende Date-Objekt enthält jedoch einen ungültigen Datumswert.

Abrufen von Werten für Zeiteinheiten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die Werte verschiedener Zeiteinheiten in einem Date-Objekt mithilfe der Eigenschaften oder Methoden der Date-Klasse abrufen. Mit jeder der folgenden Eigenschaften wird der Wert für eine bestimmte Zeiteinheit im Date-Objekt abgerufen:

- `fullYear`-Eigenschaft
- `month`-Eigenschaft (im numerischen Format von 0 für Januar bis 11 für Dezember)
- `date`-Eigenschaft (mit dem numerischen Kalendertag im Monat, im Bereich von 1 bis 31)
- `day`-Eigenschaft (der Wochentag im numerischen Format, mit dem Wert 0 für Sonntag)
- `hours`-Eigenschaft (im Bereich von 0 bis 23)
- `minutes`-Eigenschaft
- `seconds`-Eigenschaft
- `milliseconds`-Eigenschaft

  Über die Date-Klasse stehen zahlreiche Möglichkeiten zur Verfügung, um jeden dieser Werte abzurufen. Sie können den Wert für den Monat eines Date-Objekts beispielsweise auf vier verschiedene Weisen abrufen:

- mit der `month`-Eigenschaft
- mit der `getMonth()`-Methode
- mit der `monthUTC`-Eigenschaft
- mit der `getMonthUTC()`-Methode

  Alle vier Möglichkeiten sind im Wesentlichen gleich effizient, sodass Sie das Verfahren anwenden können, das sich für die entsprechende Anwendung am besten eignet.

  Alle aufgeführten Eigenschaften sind Komponenten des Date-Gesamtwerts. Der Wert der milliseconds-Eigenschaft ist beispielsweise nie größer als 999, da beim Erreichen von 1000 der Wert der seconds-Eigenschaft um 1 erhöht und der Wert der milliseconds-Eigenschaft auf 0 zurückgesetzt wird.

  Wenn Sie den Wert des Date-Objekts in Millisekunden seit dem 1. Januar 1970 (UTC) abrufen möchten, können Sie die `getTime()`-Methode verwenden. Mit dem Gegenstück, der `setTime()`-Methode, können Sie den Wert eines vorhandenen Date-Objekts in Millisekunden ab dem 1. Januar 1970 (UTC) ändern.

Berechnen von Datum und Uhrzeit

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können Datums- und Uhrzeitangaben mithilfe der Date-Klasse addieren und subtrahieren. Datumswerte werden intern in Millisekunden gespeichert, sodass Sie andere Werte in Millisekunden umrechnen sollten, bevor Sie sie zu Date-Objekten addieren oder von Date-Objekten subtrahieren.

Wenn in einer Anwendung zahlreiche Datums- und Uhrzeitberechnungen durchgeführt werden, empfiehlt es sich, Konstanten für häufig verwendete Zeiteinheiten in Millisekunden zu erstellen, wie beispielsweise die folgende Konstante:

```
public static const millisecondsPerMinute:int = 1000 * 60;
public static const millisecondsPerHour:int = 1000 * 60 * 60;
public static const millisecondsPerDay:int = 1000 * 60 * 60 * 24;
```

Nun können Datumsberechnungen mit Standardzeiteinheiten poblemlos durchgeführt werden. Mit dem folgenden Code wird ein Datumswert mithilfe der `getTime()`-Methode und der `setTime()`-Methode auf eine Stunde vor der aktuellen Zeit gesetzt:

```
var oneHourFromNow:Date = new Date();
oneHourFromNow.setTime(oneHourFromNow.getTime() + millisecondsPerHour);
```

Eine andere Möglichkeit zum Festlegen eines Datumswerts besteht darin, ein neues Date-Objekt mit einem milliseconds-Parameter zu erstellen. Mit dem folgenden Code werden beispielsweise 30 Tage zu einem Date-Objekt addiert, um dadurch ein anderes Date-Objekt zu berechnen:

```
// sets the invoice date to today's date
var invoiceDate:Date = new Date();

// adds 30 days to get the due date
var dueDate:Date = new Date(invoiceDate.getTime() + (30 * millisecondsPerDay));
```

Anschließend wird die `millisecondsPerDay`-Konstante mit 30 multipliziert, um einen Zeitraum von 30 Tagen anzugeben. Das Ergebnis wird zum Wert von `invoiceDate` addiert und zum Festlegen des Wertes von `dueDate` verwendet.

Konvertieren von Zeitangaben zwischen Zeitzonen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Datums- und Uhrzeitberechnungen können bei der Umwandlung von Datumsangaben zwischen zwei Zeitzonen eingesetzt werden. Dazu kann auch die `getTimezoneOffset()`-Methode verwendet werden, bei der der Wert in Minuten zurückgegeben wird, um die die Zeitzone des Date-Objekts von der Weltzeit abweicht. Mit der Methode wird ein Wert in Minuten zurückgegeben, da nicht alle Zeitzonen in Segmente mit vollen Stunden unterteilt sind. Bei einigen liegen halbstündliche Abweichungen zu anderen Zeitzonen vor.

Im folgenden Beispiel wird anhand des Zeitzonenunterschieds ein Date-Objekt von der Ortszeit in die Weltzeit konvertiert. Bei der Umwandlung wird zunächst der Wert für die Zeitzone in Millisekunden berechnet und der Date-Wert dann um diese Zeitangabe geändert:

```
// creates a Date in local time
var nextDay:Date = new Date("Mon May 1 2006 11:30:00 AM");

// converts the Date to UTC by adding or subtracting the time zone offset
var offsetMilliseconds:Number = nextDay.getTimezoneOffset() * 60 * 1000;
nextDay.setTime(nextDay.getTime() + offsetMilliseconds);
```

Steuern von Zeitintervallen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Entwickeln von Anwendungen mit Adobe Flash CS4 Professional können Sie auf die Zeitleiste zugreifen, mit der die entsprechende Anwendung gleichmäßig Bild für Bild durchlaufen werden kann. Bei reinen ActionScript-Projekten müssen Sie jedoch andere Verfahren zur zeitlichen Abstimmung verwenden.

Schleifen und Timer im Vergleich

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei einigen Programmiersprachen müssen Sie Zeitintervalle mit Schleifenanweisungen wie `for` oder `do..while` selbst erstellen.

Schleifenanweisungen werden in der Regel entsprechend der Geschwindigkeit des jeweiligen Computers durchgeführt, d. h. die Anwendung wird auf einigen Computern schneller ausgeführt als auf anderen. Wenn für eine Anweisung ein konstantes Zeitintervall erforderlich ist, müssen Sie eine Verknüpfung mit einem Kalenderdatum oder einer Uhrzeit erstellen. Bei vielen Anwendungen, z. B. Spielen, Animationen und Echtzeitcontrollern, werden einheitliche, die Zeit messende Timer benötigt, die auch auf unterschiedlichen Computern zu gleichen Ergebnissen kommen.

Die Timer-Klasse in ActionScript 3.0 bietet hierfür eine leistungsstarke Lösung. Unter Verwendung des Ereignismodells von ActionScript 3.0 können mit der Timer-Klasse Timer-Ereignisse ausgelöst werden, wenn ein bestimmtes Zeitintervall erreicht wird.

Timer-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Verwendung der Timer-Klasse (flash.utils.Timer) ist das empfohlene Verfahren zum Einsatz von Timerfunktionen in ActionScript 3.0. Mit dieser Klasse können Ereignisse ausgelöst werden, wenn Zeitintervalle erreicht werden.

Zum Starten eines Timers müssen Sie zunächst eine Instanz der Timer-Klasse erstellen und dann festlegen, wie oft ein Timer-Ereignis ausgelöst wird und nach wie vielen Malen es gestoppt wird.

Mit dem folgenden Code wird beispielsweise eine Timer-Instanz erstellt, mit der über einen Zeitraum von 60 Sekunden jede Sekunde ein Ereignis ausgelöst wird:

```
var oneMinuteTimer:Timer = new Timer(1000, 60);
```

Das Timer-Objekt löst jedes Mal ein TimerEvent-Objekt aus, wenn das angegebene Intervall erreicht ist. Der Ereignistyp eines TimerEvent-Objekts lautet `timer` (definiert durch die `TimerEvent.TIMER`-Konstante). Ein TimerEvent-Objekt enthält die gleichen Eigenschaften wie ein Event-Standardobjekt.

Wenn für die Timer-Instanz eine feste Anzahl von Intervallen festgelegt ist, wird auch ein `timerComplete`-Ereignis ausgelöst (definiert durch die `TimerEvent.TIMER_COMPLETE-Konstante`), wenn das letzte Intervall erreicht wird.

Es folgt eine kleine Beispielanwendung mit der Timer-Klasse:

```
package
{
    import flash.display.Sprite;
    import flash.events.TimerEvent;
    import flash.utils.Timer;

    public class ShortTimer extends Sprite
    {
        public function ShortTimer()
        {
            // creates a new five-second Timer
            var minuteTimer:Timer = new Timer(1000, 5);

            // designates listeners for the interval and completion events
            minuteTimer.addEventListener(TimerEvent.TIMER, onTick);
            minuteTimer.addEventListener(TimerEvent.TIMER_COMPLETE, onTimerComplete);

            // starts the timer ticking
            minuteTimer.start();
        }

        public function onTick(event:TimerEvent):void
        {
            // displays the tick count so far
            // The target of this event is the Timer instance itself.
            trace("tick " + event.target.currentCount);
        }

        public function onTimerComplete(event:TimerEvent):void
        {
            trace("Time's Up!");
        }
    }
}
```

Beim Erstellen der ShortTimer-Klasse wird eine Timer-Instanz erstellt, die über einen Zeitraum von 5 Sekunden einmal pro Sekunde ein Tick-Ereignis sendet. Dann werden dem Timer zwei Listener hinzugefügt: einer, der auf die einzelnen Ticks wartet, und einer, der auf das `timerComplete`-Ereignis wartet.

Anschließend wird der Timer gestartet. Ab diesem Moment wird die `onTick()`-Methode in Intervallen von einer Sekunde ausgeführt.

Über die `onTick()`-Methode wird die aktuelle Anzahl der Tick-Ereignisse angezeigt. Wenn fünf Sekunden vergangen sind, wird die `onTimerComplete()`-Methode ausgeführt. Dadurch wird angegeben, dass das Zeitintervall abgelaufen ist.

Beim Ausführen dieser Beispielanwendung werden die folgenden Zeilen in der Konsole oder im Bedienfeld „Ausgabe“ angezeigt – jeweils eine Zeile pro Sekunde:

```
tick 1
tick 2
tick 3
tick 4
tick 5
Time's Up!
```

Timerfunktionen im flash.utils-Paket

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 enthält eine Reihe von Timerfunktionen ähnlich denen in ActionScript 2.0. Diese Funktionen werden auf Paketebene im flash.utils-Paket bereitgestellt und funktionieren genauso wie in ActionScript 2.0.

Funktion	Beschreibung
`clearInterval(id:uint):void`	Bricht den angegebenen `setInterval()`-Aufruf ab
`clearTimeout(id:uint):void`	Bricht den angegebenen `setTimeout()`-Aufruf ab.
`getTimer():int`	Gibt die Anzahl der Millisekunden zurück, die seit dem Initialisieren von Adobe® Flash® Player bzw. Adobe® AIR™ vergangen sind
`setInterval(closure:Function, delay:Number, ... arguments):uint`	Führt eine Funktion in bestimmten Intervallen aus (Angabe in Millisekunden)
`setTimeout(closure:Function, delay:Number, ... arguments):uint`	Führt eine bestimmte Funktion nach der angegebenen Verzögerung aus (Angabe in Millisekunden).

Diese Funktionen wurden in ActionScript 3.0 aus Gründen der Abwärtskompatibilität beibehalten. Von der Verwendung dieser Funktionen in neuen ActionScript 3.0-Anwendungen wird abgeraten. In der Regel ist es einfacher und effizienter, in diesen Anwendungen die Timer-Klasse zu verwenden.

Beispiel für Datum und Uhrzeit: Einfache Analoguhr

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Am Beispiel einer einfachen Analoguhr werden diese beiden Verfahren zum Festlegen von Datum und Uhrzeit veranschaulicht:

- Abrufen des aktuellen Datums und der aktuellen Uhrzeit sowie Extrahieren der Werte für Stunden, Minuten und Sekunden

- Verwenden eines Timers zum Festlegen der Zeitintervalle einer Anwendung

  Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „SimpleClock“ befinden sich im Ordner „Samples/SimpleClock“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
SimpleClockApp.mxml oder SimpleClockApp.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/simpleclock/SimpleClock.as	Die Hauptanwendungsdatei.
com/example/programmingas3/simpleclock/AnalogClockFace.as	Zeichnet ein rundes Zifferblatt sowie Stunden-, Minuten- und Sekundenzeiger entsprechend der aktuellen Uhrzeit.

Definieren der SimpleClock-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Dieses Beispiel ist relativ einfach. Es empfiehlt sich jedoch, auch einfache Anwendungen gut zu strukturieren, sodass sie zu einem späteren Zeitpunkt problemlos erweitert werden können. Zu diesem Zweck werden die Start- und Zeitmessungsaufgaben in der Anwendung „SimpleClock“ mit der SimpleClock-Klasse verarbeitet. Zum Anzeigen der Zeit wird eine weitere Klasse, die AnalogClockFace-Klasse, verwendet.

Mit dem folgenden Code wird die SimpleClock-Klasse definiert und initialisiert (beachten Sie, dass SimpleClock in der Flash-Version stattdessen die Sprite-Klasse erweitert):

```
public class SimpleClock extends UIComponent
{
    /**
     * The time display component.
     */
    private var face:AnalogClockFace;

    /**
     * The Timer that acts like a heartbeat for the application.
     */
    private var ticker:Timer;
```

Die Klasse verfügt über zwei wichtige Eigenschaften:

- `face`-Eigenschaft (eine Instanz der AnalogClockFace-Klasse)
- `ticker`-Eigenschaft (eine Instanz der Timer-Klasse)

  Bei der SimpleClock-Klasse wird ein Standardkonstruktor verwendet. Mit der `initClock()`-Methode wird die Einrichtung vorgenommen, d. h. das Zifferblatt erstellt und die Timer-Instanz gestartet.

Erstellen des Zifferblatts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den folgenden Zeilen des SimpleClock-Codes wird das Zifferblatt erstellt, mit dem die Zeit angezeigt wird:

```
/**
 * Sets up a SimpleClock instance.
 */
public function initClock(faceSize:Number = 200)
{
    // creates the clock face and adds it to the display list
    face = new AnalogClockFace(Math.max(20, faceSize));
    face.init();
    addChild(face);

    // draws the initial clock display
    face.draw();
```

Die Größe des Zifferblatts kann an die `initClock()`-Methode übergeben werden. Wenn kein Wert für `faceSize` angegeben ist, wird die Standardgröße von 200 Pixel verwendet.

Dann wird das Zifferblatt initialisiert und mit der `addChild()`-Methode, die aus der DisplayObjectContainer-Klasse übernommen wurde, der Anzeigeliste hinzugefügt. Anschließend wird die `AnalogClockFace.draw()`-Methode aufgerufen, damit das Zifferblatt mit der aktuellen Uhrzeit angezeigt wird.

Starten des Timers

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Nach dem Erstellen des Zifferblatts wird mit der `initClock()`-Methode ein Timer eingerichtet:

```
    // creates a Timer that fires an event once per second
    ticker = new Timer(1000);

    // designates the onTick() method to handle Timer events
    ticker.addEventListener(TimerEvent.TIMER, onTick);

    // starts the clock ticking
    ticker.start();
```

Mit dieser Methode wird zunächst eine Timer-Instanz instanziiert, mit der einmal pro Sekunde (alle 1000 Millisekunden) ein Ereignis ausgelöst wird. Da kein zweiter `repeatCount`-Parameter an den `Timer()`-Konstruktor übergeben wird, wird der Timer unbegrenzt wiederholt.

Die `SimpleClock.onTick()`-Methode wird einmal pro Sekunde ausgeführt, wenn ein `timer`-Ereignis empfangen wird:

```
public function onTick(event:TimerEvent):void
{
    // updates the clock display
    face.draw();
}
```

Mit der `AnalogClockFace.draw()`-Methode werden das Zifferblatt und die Zeiger gezeichnet.

Anzeigen der aktuellen Uhrzeit

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem Code der AnalogClockFace-Klasse werden hauptsächlich die Anzeigeelemente des Zifferblatts eingerichtet. Nach dem Initialisieren der AnalogClockFace-Methode werden ein runder Rahmen gezeichnet und für jede Stunde eine numerische Textbeschriftung positioniert. Dann werden drei Shape-Objekte erstellt, jeweils für den Stunden-, den Minuten- und den Sekundenzeiger.

Beim Ausführen der Anwendung „SimpleClock“ wird die `AnalogClockFace.draw()`-Methode jede Sekunde aufgerufen, wie im Folgenden dargestellt:

```
/**
 * Called by the parent container when the display is being drawn.
 */
public override function draw():void
{
    // stores the current date and time in an instance variable
    currentTime = new Date();
    showTime(currentTime);
}
```

Mit dieser Methode wird die aktuelle Uhrzeit in einer Variablen gespeichert, sodass sich die Uhrzeit beim Zeichnen der Uhrzeiger nicht ändert. Dann wird die `showTime()`-Methode aufgerufen, mit der die Uhrzeiger gezeichnet werden:

```
/**
 * Displays the given Date/Time in that good old analog clock style.
 */
public function showTime(time:Date):void
{
    // gets the time values
    var seconds:uint = time.getSeconds();
    var minutes:uint = time.getMinutes();
    var hours:uint = time.getHours();

    // multiplies by 6 to get degrees
    this.secondHand.rotation = 180 + (seconds * 6);
    this.minuteHand.rotation = 180 + (minutes * 6);

    // Multiply by 30 to get basic degrees, then
    // add up to 29.5 degrees (59 * 0.5)
    // to account for the minutes.
    this.hourHand.rotation = 180 + (hours * 30) + (minutes * 0.5);
}
```

Mit dieser Methode werden zunächst die Werte für die Stunden, Minuten und Sekunden der aktuellen Uhrzeit abgerufen. Anschließend wird mit diesen Werten der Winkel für die einzelnen Uhrzeiger berechnet. Da beim Sekundenzeiger eine volle Umdrehung in 60 Sekunden erfolgt, dreht er sich pro Sekunde um 6 Grad (360/60). Auch der Minutenzeiger dreht sich pro Minute um 6 Grad.

Der Stundenzeiger wird ebenfalls jede Minute aktualisiert, sodass nach jeder Minute eine Änderung festzustellen ist. Er dreht sich pro Stunde um 30 Grad (360/12), jedoch auch pro Minute um 0,5 Grad (30 Grad geteilt durch 60 Minuten).

Kapitel 2: Arbeiten mit Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die String-Klasse enthält Methoden, die die Arbeit mit Textstrings ermöglichen. Strings sind bei der Bearbeitung vieler Objekte wichtig. Die hier beschriebenen Methoden sind nützlich bei der Bearbeitung von Strings, die beispielsweise in TextField-, StaticText-, XML-, ContextMenu- und FileReference-Objekten verwendet werden.

Bei Strings handelt es sich um Zeichenfolgen. ActionScript 3.0 unterstützt ASCII- und Unicode-Zeichen.

Verwandte Hilfethemen

String

RegExp

parseFloat()

parseInt()

Grundlagen von Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Programmierterminologie ist ein String ein Textwert – eine Folge von Buchstaben, Ziffern oder anderen Zeichen, die aneinandergereiht einen Wert ergeben. In der folgenden Codezeile wird beispielsweise eine Variable mit dem Datentyp „String“ erstellt; außerdem wird dieser Variablen ein Literalstring zugewiesen:

```
var albumName:String = "Three for the money";
```

Wie in diesem Beispiel dargestellt ist, können Sie einen Stringwert in ActionScript angeben, indem Sie Text in einfache oder doppelte Anführungszeichen setzen. Es folgen weitere Beispiele für Strings:

```
"Hello"
"555-7649"
"http://www.adobe.com/"
```

Beim Bearbeiten eines Textes in ActionScript bearbeiten Sie einen Stringwert. Zum Bearbeiten von Textwerten in ActionScript können Sie als Datentyp die String-Klasse verwenden. String-Instanzen werden häufig für Eigenschaften, Methodenparameter usw. in vielen anderen ActionScript-Klassen verwendet.

Wichtige Konzepte und Begriffe

Im Folgenden sind wichtige Begriffe aufgeführt, die im Zusammenhang mit Strings verwendet werden:

ASCII Ein System zur Darstellung von Textzeichen und Symbolen in Computerprogrammen. Das ASCII-System unterstützt das englische Alphabet mit 26 Buchstaben und eine begrenzte Gruppe zusätzlicher Zeichen.

Zeichen Die kleinste Einheit von Textdaten (ein einzelner Buchstabe oder ein einzelnes Symbol).

Verkettung Das Verbinden mehrerer Stringwerte durch Hinzufügen eines Strings am Ende eines anderen Strings zur Erstellung eines neuen Stringwerts.

Leerer String Ein String, der keinen Text, keinen Leerraum und keine anderen Zeichen enthält. Ein leerer String wird folgendermaßen angegeben: `""`. Ein leerer Stringwert unterscheidet sich von einer String-Variablen mit dem Wert „null“. Bei einer String-Variablen mit dem Wert „null“ handelt es sich um eine Variable, der keine String-Instanz zugewiesen ist. Einem leeren String ist dagegen eine Instanz mit einem Wert zugewiesen, der keine Zeichen enthält.

String Ein Textwert (Folge von Zeichen).

Stringliteral (oder „Literalstring“) Ein Stringwert, der explizit im Code als Textwert zwischen doppelten oder einfachen Anführungszeichen angegeben ist.

Teilstring Ein String, der Bestandteil eines anderen Strings ist.

Unicode Ein Standardsystem zur Darstellung von Textzeichen und Symbolen in Computerprogrammen. Das Unicode-System ermöglicht die Verwendung sämtlicher Zeichen aller auf der Welt vorkommenden Schriftsysteme.

Erstellen von Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die String-Klasse dient zum Darstellen von Stringdaten (Textdaten) in ActionScript 3.0. ActionScript-Strings unterstützen sowohl ASCII- als auch Unicode-Zeichen. Strings können am einfachsten durch Verwendung eines Stringliterals erstellt werden. Verwenden Sie doppelte gerade (`"`) oder einfache gerade (`'`) Anführungszeichen, um einen Stringliteral zu deklarieren. Die beiden folgenden Strings sind beispielsweise gleichwertig:

```
var str1:String = "hello";
var str2:String = 'hello';
```

Sie können einen String auch durch Verwendung des Operators `new` deklarieren:

```
var str1:String = new String("hello");
var str2:String = new String(str1);
var str3:String = new String();       // str3 == ""
```

Die folgenden beiden Strings sind gleichwertig:

```
var str1:String = "hello";
var str2:String = new String("hello");
```

Fügen Sie einen umgekehrten Schrägstrich (`\`) als Escape-Zeichen ein, um einfache Anführungszeichen (`'`) in einem Stringliteral zu verwenden, bei dem einfache Anführungszeichen (`'`) als Trennzeichen definiert sind. Fügen Sie ebenso einen umgekehrten Schrägstrich (`\`) als Escape-Zeichen ein, um doppelte Anführungszeichen (`"`) in einem Stringliteral zu verwenden, bei dem doppelte Anführungszeichen (`"`) als Trennzeichen definiert sind. Die folgenden beiden Strings sind gleichwertig:

```
var str1:String = "That's \"A-OK\"";
var str2:String = 'That\'s "A-OK"';
```

In Abhängigkeit davon, ob in einem Stringliteral einfache oder doppelte Anführungszeichen vorkommen, können Sie als Trennzeichen doppelte oder einfache Anführungszeichen verwenden, beispielsweise:

```
var str1:String = "ActionScript <span class='heavy'>3.0</span>";
var str2:String = '<item id="155">banana</item>';
```

Beachten Sie, dass in ActionScript zwischen einfachen geraden Anführungszeichen (`'`) und linken oder rechten einfachen typografischen Anführungszeichen (`'` oder `'`) unterschieden wird. Dies gilt auch für doppelte Anführungszeichen. Verwenden Sie gerade Anführungszeichen als Trennzeichen für Stringliterale. Achten Sie beim Einfügen von Text aus anderen Quellen in ActionScript darauf, dass die korrekten Anführungszeichen verwendet werden.

Wie in der folgenden Tabelle dargestellt, können Sie den umgekehrten Schrägstrich (`\`) als Escape-Zeichen zur Definition anderer Zeichen in Stringliteralen verwenden:

Escape-Sequenz	Zeichen
`\b`	Rücktaste
`\f`	Seitenvorschub
`\n`	Zeilenvorschub
`\r`	Wagenrücklauf
`\t`	Tab
`\u`***nnnn***	Das Unicode-Zeichen mit dem durch die Hexadezimalzahl *nnnn* spezifizierten Zeichencode. Beispielsweise ist `\u263a` das Smiley-Zeichen.
`\\x`***nn***	Das ASCII-Zeichen mit dem durch die Hexadezimalzahl *nn* spezifizierten Zeichencode.
`\'`	Einfaches Anführungszeichen
`\"`	Doppeltes Anführungszeichen
`\\`	Einfacher umgekehrter Schrägstrich

Length-Eigenschaft

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jeder String verfügt über eine `length`-Eigenschaft, die der Anzahl der Zeichen im String entspricht:

```
var str:String = "Adobe";
trace(str.length);            // output: 5
```

Wie im folgenden Beispiel dargestellt, haben leere und Null-Strings die Länge 0:

```
var str1:String = new String();
trace(str1.length);           // output: 0

str2:String = '';
trace(str2.length);           // output: 0
```

Arbeiten mit Zeichen in Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jedes Zeichen in einem String besitzt eine Indexposition (eine Ganzzahl). Die Indexposition des ersten Zeichens ist 0. Im folgenden String befindet sich das Zeichen `y` beispielsweise an Position 0 und das Zeichen `w` an Position 5:

```
"yellow"
```

Sie können einzelne Zeichen an verschiedenen Positionen eines Strings mit den Methoden `charAt()` und `charCodeAt()` überprüfen, wie im folgenden Beispiel dargestellt:

```
var str:String = "hello world!";
for (var i:int = 0; i < str.length; i++)
{
    trace(str.charAt(i), "-", str.charCodeAt(i));
}
```

Wenn Sie diesen Code ausführen, werden folgende Werte ausgegeben:

```
h - 104
e - 101
l - 108
l - 108
o - 111
- 32
w - 119
o - 111
r - 114
l - 108
d - 100
! - 33
```

Sie können zudem Zeichencodes verwenden, um einen String mit der `fromCharCode()`-Methode zu definieren, wie im folgenden Beispiel dargestellt:

```
var myStr:String = String.fromCharCode(104,101,108,108,111,32,119,111,114,108,100,33);
        // Sets myStr to "hello world!"
```

Vergleichen von Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der folgenden Operatoren können Sie Strings vergleichen: `<`, `<=`, `!=`, `==`, `=>` und `>`. Diese Operatoren können mit Bedingungsanweisungen verwendet werden, z. B. `if` und `while`, wie im folgenden Beispiel dargestellt:

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2)
{
    trace("A < a, B < b, C < c, ...");
}
```

Bei Verwendung dieser Operatoren mit Strings wird in ActionScript der Zeichencodewert der einzelnen Zeichen im String überprüft. Die Zeichen werden dabei von links nach rechts verglichen:

```
trace("A" < "B"); // true
trace("A" < "a"); // true
trace("Ab" < "az"); // true
trace("abc" < "abza"); // true
```

Verwenden Sie die Operatoren `==` und `!=`, um Strings mit anderen Strings und mit anderen Objekttypen vergleichen, wie im folgenden Beispiel dargestellt:

```
var str1:String = "1";
var str1b:String = "1";
var str2:String = "2";
trace(str1 == str1b); // true
trace(str1 == str2); // false
var total:uint = 1;
trace(str1 == total); // true
```

Stringdarstellung anderer Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können alle Objekttypen als Strings darstellen. Alle Objekte verfügen zu diesem Zweck über eine `toString()`-Methode:

```
var n:Number = 99.47;
var str:String = n.toString();
    // str == "99.47"
```

Bei Verwendung des Verkettungsoperators `+` mit einer Kombination aus String-Objekten und Objekten, die keine Strings sind, muss die `toString()`-Methode nicht eingesetzt werden. Weitere Informationen zur Verkettung finden Sie im nächsten Abschnitt.

Mit der globalen Funktion `String()` wird für ein bestimmtes Objekt der gleiche Wert zurückgegeben wie der Wert, der vom Objekt beim Aufrufen der `toString()`-Methode zurückgegeben wird.

Verketten von Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei der Verkettung von Strings werden zwei Strings sequenziell zu einem String verbunden. Sie können zwei Strings beispielsweise mit dem Operator `+` verketten:

```
var str1:String = "green";
var str2:String = "ish";
var str3:String = str1 + str2; // str3 == "greenish"
```

Das gleiche Ergebnis wird durch Verwendung des Operators `+=` erzielt, wie im folgenden Beispiel dargestellt:

```
var str:String = "green";
str += "ish"; // str == "greenish"
```

Die String-Klasse enthält zudem die `concat()`-Methode, die wie folgt verwendet werden kann:

```
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
// str4 == "Bonjour from Paris"
```

Wenn Sie den Operator `+` (oder den Operator `+=`) mit einem String-Objekt und einem Objekt, das *kein* String ist, verwenden, wird dieses Objekt in ActionScript automatisch in ein String-Objekt umgewandelt, damit der Ausdruck ausgewertet werden kann. Siehe dazu das folgende Beispiel:

```
var str:String = "Area = ";
var area:Number = Math.PI * Math.pow(3, 2);
str = str + area; // str == "Area = 28.274333882308138"
```

Sie können jedoch mithilfe von Gruppierungsklammern Kontext für den Operator + angeben, wie im folgenden Beispiel dargestellt:

```
trace("Total: $" + 4.55 + 1.45); // output: Total: $4.551.45
trace("Total: $" + (4.55 + 1.45)); // output: Total: $6
```

Suchen von Teilstrings und Mustern in Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Teilstrings sind Zeichenfolgen innerhalb eines Strings. Der String „`abc`" beispielsweise hat die folgenden Teilstrings: „", „`a`", „`ab`", „`abc`", „`b`", „`bc`", „`c`". Mithilfe von ActionScript-Methoden können Sie die Teilstrings eines Strings suchen.

Muster sind in ActionScript durch Strings oder reguläre Ausdrücke definiert. Der folgende reguläre Ausdruck legt beispielsweise ein bestimmtes Muster fest: die Buchstaben A, B und C, gefolgt von einer Ziffer (die Schrägstriche werden in regulären Ausdrücken als Trennzeichen verwendet):

```
/ABC\d/
```

ActionScript enthält Methoden zum Suchen von Mustern in Strings sowie zum Ersetzen der gefundenen Entsprechungen durch Teilstrings. Diese Methoden werden in den folgenden Abschnitten beschrieben.

Mit regulären Ausdrücken können komplizierte Muster definiert werden. Weitere Informationen finden Sie unter „Verwenden von regulären Ausdrücken" auf Seite 81.

Suchen eines Teilstrings nach Zeichenposition

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Methoden `substr()` und `substring()` sind sehr ähnlich. Beide geben den Teilstring eines Strings zurück. Bei erwarten zwei Parameter. Bei beiden Methoden ist der erste Parameter die Position des Startzeichens des jeweiligen Strings. Allerdings ist der zweite Parameter bei der `substr()`-Methode die *Länge* des zurückzugebenden Teilstrings und bei der `substring()`-Methode die Position des Zeichens am *Ende* des Teilstrings (das selbst nicht mehr Bestandteil des zurückgegebenen Strings ist). Im folgenden Beispiel ist der Unterschied zwischen beiden Methoden dargestellt:

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.substr(11,15)); // output: Paris, Texas!!!
trace(str.substring(11,15)); // output: Pari
```

Die `slice()`-Methode ähnelt der `substring()`-Methode. Wenn zwei nicht negative Ganzzahlen als Parameter angegeben werden, ist die Funktionsweise identisch. Bei der `slice()`-Methode können jedoch negative Ganzzahlen als Parameter verwendet werden. Die Zeichenposition wird dann vom Ende des Strings gezählt, wie im folgenden Beispiel dargestellt:

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.slice(11,15)); // output: Pari
trace(str.slice(-3,-1)); // output: !!
trace(str.slice(-3,26)); // output: !!!
trace(str.slice(-3,str.length)); // output: !!!
trace(str.slice(-8,-3)); // output: Texas
```

Bei der `slice()`-Methode können Sie nicht negative und negative Ganzzahlen als Parameter kombinieren.

Suchen der Zeichenposition eines übereinstimmenden Teilstrings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den Methoden `indexOf()` und `lastIndexOf()` können Sie übereinstimmende Teilstrings in einem String suchen, wie im folgenden Beispiel dargestellt:

```
var str:String = "The moon, the stars, the sea, the land";
trace(str.indexOf("the")); // output: 10
```

Bei der `indexOf()`-Methode wird die Groß- und Kleinschreibung beachtet.

Sie können einen zweiten Parameter festlegen, um die Indexposition im String anzugeben, ab der die Suche gestartet wird:

```
var str:String = "The moon, the stars, the sea, the land"
trace(str.indexOf("the", 11)); // output: 21
```

Mit der `lastIndexOf()`-Methode wird das letzte Vorkommen eines Teilstrings in einem String gesucht:

```
var str:String = "The moon, the stars, the sea, the land"
trace(str.lastIndexOf("the")); // output: 30
```

Wenn Sie bei der `lastIndexOf()`-Methode einen zweiten Parameter einfügen, wird die Suche ab der Indexposition des Strings rückwärts (von rechts nach links) durchgeführt:

```
var str:String = "The moon, the stars, the sea, the land"
trace(str.lastIndexOf("the", 29)); // output: 21
```

Erstellen eines durch Trennzeichen unterteilten Arrays von Teilstrings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der `split()`-Methode können Sie ein Array aus Teilstrings erstellen, die anhand eines Trennzeichens unterteilt sind. Sie können beispielsweise einen String, der durch Kommas oder Tabulatoren getrennt ist, in mehrere Strings unterteilen.

Im folgenden Beispiel ist dargestellt, wie ein Array mit dem Und-Zeichen (&) als Trennzeichen in Teilstrings unterteilt wird:

```
var queryStr:String = "first=joe&last=cheng&title=manager&StartDate=3/6/65";
var params:Array = queryStr.split("&", 2); // params == ["first=joe","last=cheng"]
```

Der optionale zweite Parameter der `split()`-Methode definiert die maximale Größe des zurückgegebenen Arrays.

Sie können auch einen regulären Ausdruck als Trennzeichen verwenden:

```
var str:String = "Give me\t5."
var a:Array = str.split(/\s+/); // a == ["Give","me","5."]
```

Weitere Informationen finden Sie unter „Verwenden von regulären Ausdrücken“ auf Seite 81 und im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Suchen von Mustern in Strings und Ersetzen von Teilstrings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die String-Klasse enthält folgende Methoden zum Verwenden von Mustern in Strings:

- Verwenden Sie die Methoden `match()` und `search()` zum Suchen von Teilstrings, die einem Muster entsprechen.
- Verwenden Sie die `replace()`-Methode, um Teilstrings zu suchen, die einem Muster entsprechen, und sie durch den angegebenen Teilstring zu ersetzen.

Diese Methoden werden in den folgenden Abschnitten beschrieben.

Sie können die bei diesen Methoden verwendeten Muster mithilfe von Strings oder regulären Ausdrücken definieren. Weitere Informationen zu regulären Ausdrücken finden Sie unter „Verwenden von regulären Ausdrücken“ auf Seite 81.

Suchen von übereinstimmenden Teilstrings

Die `search()`-Methode gibt die Indexposition des ersten Teilstrings zurück, der einem bestimmten Muster entspricht, wie im folgenden Beispiel dargestellt:

```
var str:String = "The more the merrier.";
// (This search is case-sensitive.)
trace(str.search("the")); // output: 9
```

Sie können auch mithilfe regulärer Ausdrücke das entsprechende Muster festlegen:

```
var pattern:RegExp = /the/i;
var str:String = "The more the merrier.";
trace(str.search(pattern)); // 0
```

Von der `trace()`-Methode wird 0 zurückgegeben, da sich das erste Zeichen des Strings an Indexposition 0 befindet. Im regulären Ausdruck ist das `i`-Flag gesetzt, sodass die Groß- und Kleinschreibung bei der Suche nicht beachtet wird.

Die `search()`-Methode findet nur eine Entsprechung und gibt die zugehörige Indexposition zurück, auch wenn im regulären Ausdruck das `g`-Flag (global) gesetzt ist.

Im folgenden Beispiel ist ein komplizierterer regulärer Ausdruck dargestellt, der in doppelte Anführungszeichen eingeschlossene Strings findet:

```
var pattern:RegExp = /"[^"]*"/;
var str:String = "The \"more\" the merrier.";
trace(str.search(pattern)); // output: 4

str = "The \"more the merrier.";
trace(str.search(pattern)); // output: -1
// (Indicates no match, since there is no closing double quotation mark.)
```

Mit der `match()`-Methode wird auf ähnliche Weise ein übereinstimmender Teilstring gesucht. Bei Verwendung des global-Flags in regulären Ausdrücken (siehe folgendes Beispiel) wird mit der `match()`-Methode jedoch ein Array übereinstimmender Teilstrings zurückgegeben:

```
var str:String = "bob@example.com, omar@example.org";
var pattern:RegExp = /\w*@\w*\.[org|com]+/g;
var results:Array = str.match(pattern);
```

Das `results`-Array enthält folgende Werte:

```
["bob@example.com","omar@example.org"]
```

Weitere Informationen zu regulären Ausdrücken finden Sie unter „Verwenden von regulären Ausdrücken“ auf Seite 81.

Ersetzen von übereinstimmenden Teilstrings

Mithilfe der `replace()`-Methode können Sie ein bestimmtes Muster in einem String suchen und Übereinstimmungen durch den angegebenen String ersetzen, wie im folgenden Beispiel dargestellt:

```
var str:String = "She sells seashells by the seashore.";
var pattern:RegExp = /sh/gi;
trace(str.replace(pattern, "sch")); //sche sells seaschells by the seaschore.
```

Bei den übereinstimmenden Strings in diesem Beispiel wird die Groß- und Kleinschreibung nicht berücksichtigt, da im regulären Ausdruck das `i`-Flag (`ignoreCase`) gesetzt ist. Zudem werden alle Entsprechungen ersetzt, da das `g`-Flag (`global`) gesetzt ist. Weitere Informationen finden Sie unter „Verwenden von regulären Ausdrücken“ auf Seite 81.

Sie können die folgenden `$`-Ersetzungscodes im Ersetzungsstring einfügen. Anstelle des `$`-Ersetzungscodes wird jeweils der in der folgenden Tabelle aufgeführte Ersetzungstext eingefügt:

$-Code	**Ersetzungstext**
`$$`	$
`$&`	Der übereinstimmende Teilstring.
`$` `	Der Teil des Strings, der vor dem übereinstimmenden Teilstring steht. Für diesen Code wird das nach links gerichtete gerade Anführungszeichen (`) und nicht das gerade einfache Anführungszeichen (') oder das linke einfache typografische Anführungszeichen (') verwendet.
`$'`	Der Teil des Strings, der nach dem übereinstimmenden Teilstring steht. Für diesen Code wird das gerade einfache Anführungszeichen (') verwendet.
$*n*	Die *n*. zwischengespeicherte, in Klammern eingeschlossene Gruppe, wobei n für eine Ziffer zwischen 1 und 9 steht und nach $n keine weitere Dezimalziffer folgt.
$*nn*	Die *nn*. erfasste, in Klammern eingeschlossene übereinstimmende Gruppe, wobei *nn* für eine zweistellige Dezimalzahl zwischen 01 und 99 steht. Wenn der *nn*. erfasste Wert nicht definiert ist, ist der Ersetzungstext ein leerer String.

So zeigt das folgende Beispiel die Verwendung der Ersetzungscodes `$2` und `$1`, die die erste und zweite erfasste, übereinstimmende Gruppe repräsentieren:

```
var str:String = "flip-flop";
var pattern:RegExp = /(\w+)-(\w+)/g;
trace(str.replace(pattern, "$2-$1")); // flop-flip
```

Sie können als zweiten Parameter der `replace()`-Methode auch eine Funktion verwenden. Der übereinstimmende Text wird durch den zurückgegebenen Wert der Funktion ersetzt.

```
var str:String = "Now only $9.95!";
var price:RegExp = /\$([\d,]+.\d+)+/i;
trace(str.replace(price, usdToEuro));

function usdToEuro(matchedSubstring:String,  capturedMatch1:String,  index:int,
str:String):String
{
    var usd:String = capturedMatch1;
    usd = usd.replace(",", "");
    var exchangeRate:Number = 0.853690;
    var euro:Number = parseFloat(usd) * exchangeRate;
    const euroSymbol:String = String.fromCharCode(8364);
    return euro.toFixed(2) + " " + euroSymbol;
}
```

Wenn eine Funktion als zweiter Parameter der `replace()`-Methode verwendet wird, werden die folgenden Argumente an die Funktion übergeben:

- Der übereinstimmende Teil des Strings
- Alle übereinstimmenden zwischengespeicherten Gruppen. Die Anzahl der auf diese Weise übergebenen Argumente hängt von der Anzahl der in Klammern eingeschlossenen übereinstimmenden Gruppen ab. Um die Anzahl der Übereinstimmungen mit einer in Klammern eingeschlossene Gruppe zu bestimmen, verwenden Sie `arguments.length - 3` innerhalb des Funktionscodes.
- Die Indexposition im String, an der die Übereinstimmung beginnt.
- Der vollständige String.

Umwandeln der Groß- und Kleinschreibung von Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie im folgenden Beispiel dargestellt, konvertieren die Methoden `toLowerCase()` und `toUpperCase()` Buchstaben in einem String in Kleinbuchstaben bzw. Großbuchstaben:

```
var str:String = "Dr. Bob Roberts, #9."
trace(str.toLowerCase()); // dr. bob roberts, #9.
trace(str.toUpperCase()); // DR. BOB ROBERTS, #9.
```

Der Quellstring wird beim Ausführen dieser Methoden nicht geändert. Verwenden Sie den folgendem Code, um den Ausgangsstring umzuwandeln:

```
str = str.toUpperCase();
```

Diese Methoden können auch bei Sonderzeichen und nicht nur bei a bis z und A bis Z verwendet werden:

```
var str:String = "José Barça";
trace(str.toUpperCase(), str.toLowerCase()); // JOSÉ BARÇA josé barça
```

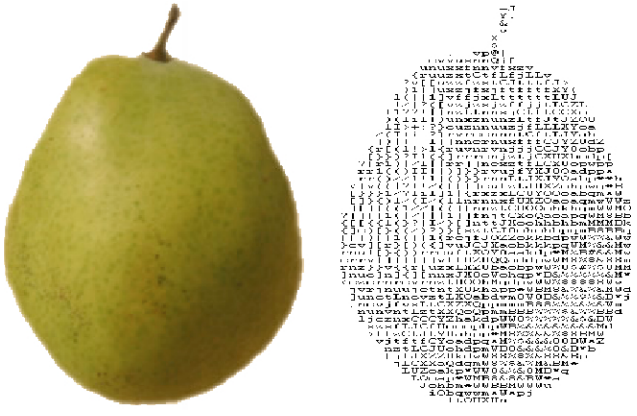
Strings-Beispiel: ASCII-Grafik

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Im folgenden Beispiel einer ASCII-Grafik werden mehrere Funktionen bei der Verwendung der String-Klasse in ActionScript 3.0 erläutert, einschließlich folgender Funktionen:

- Mithilfe der `split()`-Methode der String-Klasse werden Werte eines durch Zeichen getrennten Strings extrahiert (Bilddaten in einer durch Tabulatoren getrennten Textdatei).
- Mehrere Verfahren zur Bearbeitung von Strings, einschließlich `split()`, Verkettung und Extraktion von Teilstrings mit `substring()` und `substr()` werden zur Großschreibung des ersten Buchstabens jedes Wortes in Bildtiteln verwendet.
- Mithilfe der `getCharAt()`-Methode wird ein einzelnes Zeichen eines Strings abgerufen (um das ASCII-Zeichen zu ermitteln, das einem Graustufen-Bitmapwert entspricht).
- Mithilfe der Stringverkettung wird Zeichen für Zeichen die ASCII-Grafik-Darstellung eines Bilds erstellt.

Der Begriff *ASCII-Grafik* bezieht sich auf Textdarstellungen eines Bilds, bei dem das Bild durch ein Raster aus Schriftzeichen fester Breite (z. B. Courier New) erstellt wird. Das folgende Bild ist ein Beispiel für eine in der Anwendung erstellte ASCII-Grafik:



Die ASCII-Grafikversion des Bilds ist rechts dargestellt.

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „ASCIIArt“ befinden sich im Ordner „Samples/AsciiArt“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
AsciiArtApp.mxml oder AsciiArtApp.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/asciiArt/AsciiArtBuilder.as	Die Klasse mit den Hauptfunktionen der Anwendung, einschließlich Extrahieren der Bildmetadaten aus einer Textdatei, Laden der Bilder und Verwalten des Vorgangs zum Konvertieren von Bildern in Text.
com/example/programmingas3/asciiArt/BitmapToAsciiConverter.as	Eine Klasse mit der `parseBitmapData()`-Methode zum Konvertieren von Bilddaten in Strings.

Datei	Beschreibung
com/example/programmingas3/asciiArt/Image.as	Eine Klasse, die ein geladenes Bitmapbild darstellt.
com/example/programmingas3/asciiArt/ImageInfo.as	Eine Klasse mit den Metadaten einer ASCII-Grafik (z. B. Titel oder Bilddatei-URL).
image/	Ein Ordner mit den in der Anwendung verwendeten Bildern.
txt/ImageData.txt	Eine durch Tabulatoren getrennte Textdatei mit Informationen zu den Bildern, die von der Anwendung geladen werden.

Extrahieren von durch Tabulatoren getrennten Werten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In diesem Beispiel sind entsprechend der gängigen Praxis die Anwendungsdaten getrennt von der Anwendung gespeichert. Auf diese Weise muss die SWF-Datei nicht neu erstellt werden, wenn sich die Daten ändern (z. B. beim Hinzufügen eines weiteren Bilds oder Ändern eines Bildtitels). In diesem Fall sind die Bildmetadaten, darunter der Bildtitel, die URL der Bilddatei und einige Werte zum Bearbeiten des Bilds, in einer Textdatei gespeichert (Datei „txt/ImageData.txt“ im Projekt). Die Textdatei enthält Folgendes:

```
FILENAMETITLEWHITE_THRESHHOLDBLACK_THRESHHOLD
FruitBasket.jpgPear, apple, orange, and bananad810
Banana.jpgA picture of a bananaC820
Orange.jpgorangeFF20
Apple.jpgpicture of an apple6E10
```

Die Datei weist ein spezifisches, durch Tabulatoren getrenntes Format auf. Die erste Zeile ist eine Kopfzeile. Die restlichen Zeilen enthalten die folgenden Daten für jede zu ladende Bitmap:

- Den Dateinamen der Bitmap.
- Den Anzeigenamen der Bitmap.
- Die Schwellenwerte für die Weiß- und Schwarztöne der Bitmap-Dateien. Dabei handelt es sich um hexadezimale Werte, oberhalb bzw. unterhalb denen ein Pixel als vollständig weiß bzw. vollständig schwarz eingeordnet wird.

Nach dem Starten der Anwendung wird die AsciiArtBuilder-Klasse geladen. Der Inhalt der Textdatei wird mit dem folgenden Code aus der `parseImageInfo()`-Methode der AsciiArtBuilder-Klasse analysiert, um die anzuzeigenden Bilder zu erstellen:

```
var lines:Array = _imageInfoLoader.data.split("\n");
var numLines:uint = lines.length;
for (var i:uint = 1; i < numLines; i++)
{
    var imageInfoRaw:String = lines[i];
    ...
    if (imageInfoRaw.length > 0)
    {
        // Create a new image info record and add it to the array of image info.
        var imageInfo:ImageInfo = new ImageInfo();

        // Split the current line into values (separated by tab (\t)
        // characters) and extract the individual properties:
        var imageProperties:Array = imageInfoRaw.split("\t");
        imageInfo.fileName = imageProperties[0];
        imageInfo.title = normalizeTitle(imageProperties[1]);
        imageInfo.whiteThreshold = parseInt(imageProperties[2], 16);
        imageInfo.blackThreshold = parseInt(imageProperties[3], 16);
        result.push(imageInfo);
    }
}
```

Der gesamte Inhalt der Textdatei befindet sich in einer einzigen String-Instanz, der Eigenschaft `_imageInfoLoader.data`. Über die `split()`-Methode mit dem Zeilenvorschubzeichen (`\n`) als Parameter wird die String-Instanz in ein Array (`lines`) unterteilt, dessen Elemente die einzelnen Zeilen der Textdatei darstellen. Die einzelnen Zeilen werden in einer Schleife verarbeitet (mit Ausnahme der ersten Zeile, die nur Kopfzeilen und keinen Inhalt enthält). Innerhalb der Schleife wird der Inhalt jeder Zeile wieder mit der `split()`-Methode in eine Wertegruppe unterteilt (das Array-Objekt `imageProperties`). Als Parameter der `split()`-Methode wird in diesem Fall das Tabulatorzeichen (`\t`) verwendet, da die Werte in jeder Zeile durch Tabulatoren getrennt sind.

Verwenden von String-Methoden zum Normalisieren von Bildtiteln

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In einer Entwurfsentscheidung für diese Anwendung wurde festgelegt, dass alle Bildtitel im Standardformat angezeigt werden, d. h. Großschreibung des jeweils ersten Buchstabens jedes Worts (mit Ausnahme einiger Wörter, die in englischen Titeln normalerweise nicht in Großbuchstaben geschrieben werden). Die in der Textdatei enthaltenen Titel werden beim Extrahieren aus der Textdatei von der Anwendung formatiert.

Im vorherigen Code wird als Bestandteil der Extraktion einzelner Werte der Bildmetadaten die folgende Codezeile verwendet:

```
imageInfo.title = normalizeTitle(imageProperties[1]);
```

Im Code dieser Funktion wird der Bildtitel aus der Textdatei über die `normalizeTitle()`-Methode übergeben und dann im ImageInfo-Objekt gespeichert:

```
private function normalizeTitle(title:String):String
{
    var words:Array = title.split(" ");
    var len:uint = words.length;
    for (var i:uint; i < len; i++)
    {
        words[i] = capitalizeFirstLetter(words[i]);
    }

    return words.join(" ");
}
```

Mit dieser Methode wird der Titel mithilfe der `split()`-Methode in einzelne (durch Leerzeichen getrennte) Wörter unterteilt. Alle Wörter werden der `capitalizeFirstLetter()`-Methode übergeben und dann über die `join()`-Methode der Array-Klasse wieder zu einem einzelnen String zusammengesetzt.

Wie der Name vermuten lässt, erfolgt die Großschreibung des ersten Buchstabens der einzelnen Wörter über die `capitalizeFirstLetter()`-Methode:

```
    /**
     * Capitalizes the first letter of a single word, unless it's one of
     * a set of words that are normally not capitalized in English.
     */
    private function capitalizeFirstLetter(word:String):String
    {
        switch (word)
        {
            case "and":
            case "the":
            case "in":
            case "an":
            case "or":
            case "at":
            case "of":
            case "a":
                // Don't do anything to these words.
                break;
            default:
                // For any other word, capitalize the first character.
                var firstLetter:String = word.substr(0, 1);
                firstLetter = firstLetter.toUpperCase();
                var otherLetters:String = word.substring(1);
                word = firstLetter + otherLetters;
        }
        return word;
    }
```

Im Englischen wird das erste Zeichen eines Worts in einem Titel *nicht* großgeschrieben, wenn es sich um eines der folgenden Wörter handelt: „and“ „the“, „in“, „an“, „or“, „at“, „of“ oder „a“. (Dies ist eine vereinfachte Version der Regeln.) Im Code wird zuerst mithilfe einer `switch`-Anweisung überprüft, ob es sich bei einem Wort um eines der Wörter handelt, die nicht großgeschrieben werden sollen. Wenn dies der Fall ist, wird die `switch`-Anweisung einfach übersprungen. Wenn ein Wort jedoch großgeschrieben werden muss, erfolgt dies in mehreren Folgeschritten:

1. Der erste Buchstabe des Worts wird mit `substr(0, 1)` extrahiert. Dabei wird ein Teilstring ab dem Zeichen bei Indexposition 0 extrahiert (der erste Buchstabe im String, wie durch den ersten Parameter `0` angegeben). Der Teilstring hat die Länge eines Zeichens (durch den zweiten Parameter `1` angegeben).

2 Dieses Zeichen wird mithilfe der `toUpperCase()`-Methode in einen Großbuchstaben umgewandelt.

3 Die restlichen Zeichen des Worts werden mit `substring(1)` extrahiert. Dabei wird ein Teilstring ab Indexposition 1 (der zweite Buchstabe) bis zum Ende des Strings (keine Angabe des zweiten Parameters für die `substring()`-Methode) extrahiert.

4 Das endgültige Wort wird durch Kombinieren des nun großgeschriebenen ersten Buchstabens mit den restlichen Buchstaben über folgende Stringverkettung erstellt: `firstLetter + otherLetters`

Erzeugen des Textes für die ASCII-Grafik

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die BitmapToAsciiConverter-Klasse enthält die Funktionen zum Konvertieren eines Bitmapbilds in die entsprechende ASCII-Textdarstellung. Dieser Vorgang wird über die `parseBitmapData()`-Methode ausgeführt, die im Folgenden teilweise aufgeführt ist:

```
var result:String = "";

// Loop through the rows of pixels top to bottom:
for (var y:uint = 0; y < _data.height; y += verticalResolution)
{
    // Within each row, loop through pixels left to right:
    for (var x:uint = 0; x < _data.width; x += horizontalResolution)
    {
        ...

        // Convert the gray value in the 0-255 range to a value
        // in the 0-64 range (since that's the number of "shades of
        // gray" in the set of available characters):
        index = Math.floor(grayVal / 4);
        result += palette.charAt(index);
    }
    result += "\n";
}
return result;
```

Mit diesem Code wird zunächst die String-Instanz `result` definiert, in der die ASCII-Grafikversion des Bitmapbilds erstellt wird. Dann werden die einzelnen Pixel des Quellbitmapbilds durchlaufen. Mit mehreren Verfahren zur Farbbearbeitung (hier aus Platzgründen weggelassen) werden die Rot-, Grün- und Blaufarbwerte eines Pixels in einen Graustufenwert (Zahl von 0 bis 255) konvertiert. Dieser Wert wird dann (wie dargestellt) durch 4 dividiert, sodass er in einen Wert konvertiert wird, der im Bereich zwischen 0 und 63 liegt, und dann in der Variablen `index` gespeichert. (Der Bereich 0-63 wird verwendet, da der Bereich der verfügbaren ASCII-Zeichen in dieser Anwendung 64 Werte umfasst.) Der Bereich der Zeichen ist in der BitmapToAsciiConverter-Klasse als String-Instanz definiert:

```
// The characters are in order from darkest to lightest, so that their
// position (index) in the string corresponds to a relative color value
// (0 = black).
private static const palette:String =
"@#$%&8BMW*mwqpdbkhaoQ0OZXYUJCLtfjzxnuvcr[]{}1()|/?Il!i><+_~-;,. ";
```

Da durch die `index`-Variable festgelegt ist, welches ASCII-Zeichen im Bereich dem aktuellen Pixel im Bitmapbild entspricht, wird das entsprechende Zeichen mit der `charAt()`-Methode aus dem `palette`-String abgerufen. Es wird dann mit dem Verkettungszuweisungsoperator (`+=`) an die String-Instanz `result` angehängt. Zusätzlich wird am Ende jeder Pixelzeile ein Zeilenvorschubzeichen mit dem Ende des Strings `result` verkettet, sodass die Zeile jeweils umgebrochen und eine neue Zeile mit „Buchstabenpixeln“ erstellt wird.

Kapitel 3: Verwenden von Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit Arrays können Sie mehrere Werte in einer einzelnen Datenstruktur speichern. Sie können einfache indizierte Arrays verwenden, in denen Werte mit Indizes mit festgelegten Ordnungszahlen gespeichert werden, oder komplexe assoziative Arrays, in denen Werte mit beliebigen Schlüsseln gespeichert werden. Arrays können darüber hinaus mehrdimensional sein und Elemente enthalten, die selbst Arrays sind. Außerdem können Sie einen Vektor für ein Array verwenden, dessen Elemente sämtlich Instanzen desselben Datentyps sind.

Verwandte Hilfethemen

Array

Vektor

Grundlagen von Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Programmieren müssen Sie häufig anstelle eines einzelnen Objekts eine Gruppe von Objekten bearbeiten. In einer Anwendung für einen Audio-Player kann es beispielsweise wünschenswert sein, eine Wiedergabeliste für Musiktitel zu erstellen. Es wäre unpraktisch, eine separate Variable für jeden Musiktitel in der Liste erstellen zu müssen. Stattdessen sollten sich alle Musiktitel-Objekte zusammen in einem Paket befinden, sodass Sie sie als Gruppe verwenden können.

Bei einem Array handelt es sich um ein Programmierelement, das als Container für eine Gruppe von Objekten dient, z. B. für eine Wiedergabeliste mit Musiktiteln. In der Regel sind alle Elemente in einem Array Instanzen derselben Klasse; dies ist in ActionScript jedoch nicht zwingend erforderlich. Die einzelnen Objekte in einem Array werden als *Elemente* bezeichnet. Sie können sich ein Array als Aktenschublade für Variablen vorstellen. Variablen können dem Array als Elemente hinzugefügt werden, wie Ordner in die Aktenschublade gelegt werden. Sie können das Array wie eine einzelne Variable verwenden (so wie Sie beispielsweise die ganze Schublade an einen anderen Ort tragen können). Sie haben die Möglichkeit, die Variablen als Gruppe zu bearbeiten (vergleichbar mit dem Suchen nach Informationen, indem Sie die Ordner nacheinander durchblättern). Außerdem ist der Zugriff auf einzelne Variablen möglich (Öffnen der Schublade und Auswahl eines einzelnen Ordners).

Beispiel: Sie erstellen eine Anwendung für einen Audio-Player, in der ein Benutzer mehrere Musiktitel auswählen und einer Wiedergabeliste hinzufügen kann. Ihr ActionScript-Code enthält eine Methode namens `addSongsToPlaylist()`, der als Parameter ein einzelnes Array übergeben wird. Unabhängig von der Anzahl der zur Wiedergabeliste hinzugefügten Musiktitel (einige, viele oder nur ein Musiktitel) muss die `addSongsToPlaylist()`-Methode nur ein Mal aufgerufen werden. Dabei wird das Array mit den Song-Objekten übergeben. In der `addSongsToPlaylist()`-Methode können die Elemente des Arrays (die Musiktitel) in einer Schleife ablaufen und der Wiedergabeliste hinzugefügt werden.

Am häufigsten tritt in ActionScript das sogenannte *indizierte Array* auf. Bei einem indizierten Array wird jedes Element an einer nummerierten Position gespeichert (*Index* genannt). Der Zugriff auf die Elemente erfolgt über die Nummer, die die Funktion einer Adresse hat. Indizierte Arrays können für die meisten Programmieranforderungen eingesetzt werden. Zum Repräsentieren eines indizierten Arrays wird häufig die Array-Klasse verwendet.

Oft dient ein indiziertes Array zum Speichern von mehreren Elementen desselben Typs (also von Objekten, die Instanzen derselben Klasse sind). Die Array-Klasse bietet keine Möglichkeit, den Typ der in ihr enthaltenen Elemente einzuschränken. Mit der Vector-Klasse kann ein indiziertes Array erstellt werden, in dem alle Elemente denselben Typ aufweisen. Bei Verwendung einer Vector-Instanz anstelle einer Array-Instanz können auch Leistungsverbesserungen und andere Vorteile erzielt werden. Die Vector-Klasse ist verfügbar ab Flash Player 10 und Adobe AIR 1.5.

Eine besondere Art eines indizierten Arrays ist ein *mehrdimensionales Array*. Bei einem mehrdimensionalen Array handelt es sich um ein indiziertes Array, dessen Elemente ebenfalls indizierte Arrays sind (die wiederum andere Elemente enthalten).

Ein anderer Array-Typ ist das *assoziative Array*, in dem einzelne Elemente mit einem *Stringschlüssel* und nicht mit einer numerischen Indexposition identifiziert werden. Schließlich gehört zu ActionScript 3.0 noch die Dictionary-Klasse, die ein *Wörterbuch* repräsentiert. Ein Wörterbuch ist ein Array, in dem Sie jeden Objekttyp als Schlüssel verwenden können, um verschiedene Elemente zu unterscheiden.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die Ihnen beim Programmieren von Array- und Vektorverarbeitungsroutinen begegnen:

Array Ein Objekt, das als Container zum Gruppieren mehrerer Objekte dient.

Array-Zugriffsoperator ([]) Eckige Klammern, die eine Indexposition oder einen Schlüssel umgeben, der ein Array-Element eindeutig identifiziert. Diese Syntax wird nach einem Array-Variablennamen verwendet, um statt des ganzen Arrays ein einzelnes Element des Arrays anzugeben.

Assoziatives Array Ein Array mit Stringschlüsseln zum Identifizieren der einzelnen Elemente.

Basistyp Der Datentyp der Objekte, die in einer Vector-Instanz gespeichert werden können.

Wörterbuch Ein Array, dessen Elemente sich aus Objektpaaren zusammensetzen, die als Schlüssel und Wert bezeichnet werden. Der Schlüssel wird anstelle eines numerischen Indexes zum Identifizieren eines einzelnen Elements verwendet.

Element Ein einzelnes Objekt in einem Array.

Index Die numerische „Adresse“ zum Identifizieren eines einzelnen Elements in einem indizierten Array.

Indiziertes Array Der Standardtyp für Arrays, in dem jedes Element an einer nummerierten Position gespeichert wird und bei dem die einzelnen Elemente anhand der Nummer (des Indexes) identifiziert werden.

Schlüssel Der String oder das Objekt zum Identifizieren eines einzelnen Elements in einem assoziativen Array oder einem Wörterbuch.

Mehrdimensionales Array Ein Array mit Elementen, die selbst Arrays und keine Einzelwerte sind.

T Die Standardkonvention, mit der in dieser Dokumentation der Basistyp einer Vector-Instanz angegeben wird. Die T-Konvention repräsentiert einen Klassennamen, wie in der Beschreibung des Type-Parameters gezeigt. („T“ steht für „Typ“, wie in „Datentyp“.).

Type-Parameter Die Syntax, mit der der Vector-Basistyp im Vector-Klassennamen angegeben wird (der Datentyp der gespeicherten Objekte). Die Syntax besteht aus einem Punkt (`.`) gefolgt vom Namen des Datentyps in spitzen Klammern (`<>`). Daraus ergibt sich folgende Angabe: `Vector.<T>`. In dieser Dokumentation wird die im Type-Parameter angegebene Klasse generisch durch `T` repräsentiert.

Vektor Ein Array-Typ, bei dem alle Elemente Instanzen desselben Datentyps sind.

Indizierte Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In indizierten Arrays wird eine Reihe aus einem oder mehreren Werten gespeichert, die so angeordnet sind, dass auf jeden Wert über eine vorzeichenlose Ganzzahl zugegriffen werden kann. Die erste Indexposition ist immer die Zahl 0. Mit jedem dem Array hinzugefügten Element wird der Index um den Wert 1 erhöht. In ActionScript 3.0 werden zwei Klassen als indizierte Arrays verwendet: die Array-Klasse und die Vector-Klasse.

Bei indizierten Arrays wird für die Indexnummer eine vorzeichenlose 32-Bit-Ganzzahl verwendet. Die maximale Größe eines indizierten Arrays beträgt 2^{32} - 1 oder 4.294.967.295. Beim Versuch, ein Array zu erstellen, das größer ist als die maximale Größe, wird zur Laufzeit eine Fehlermeldung ausgegeben.

Zum Zugriff auf ein einzelnes Element in einem indizierten Array verwenden Sie den Array-Zugriffsoperator (`[]`), um die Indexposition des gewünschten Elements anzugeben. Der folgende Code repräsentiert beispielsweise das erste Element (das Element an Indexposition 0) in einem indizierten Array namens `songTitles`:

```
songTitles[0]
```

Aus der Kombination des Array-Variablennamens gefolgt von der Indexposition in eckigen Klammern ergibt sich ein einzelner Bezeichner. (Das heißt, der Bezeichner kann genau wie jeder Variablenname verwendet werden.) Sie können einem Element eines indizierten Arrays einen Wert zuweisen, indem Sie den Namen und die Indexposition auf der linken Seite einer Zuweisungsanweisung verwenden:

```
songTitles[1] = "Symphony No. 5 in D minor";
```

Gleichermaßen können Sie den Wert eines Elements in einem indizierten Array abrufen, indem Sie den Namen und die Indexposition auf der rechten Seite einer Zuweisungsanweisung verwenden:

```
var nextSong:String = songTitles[2];
```

Die eckigen Klammern können anstelle eines expliziten Wertes auch eine Variable enthalten. (Die Variable muss eine nichtnegative Ganzzahl enthalten, wie beispielsweise „uint“, „positive int“ oder eine Number-Instanz mit einer positiven Ganzzahl). Dieses Verfahren wird häufig verwendet, um die Elemente in einem indizierten Array in einer Schleife zu durchlaufen und einen Vorgang für einige oder alle Elemente auszuführen. Im folgenden Codebeispiel wird dieses Verfahren veranschaulicht. Der Code verwendet eine Schleife für den Zugriff auf jeden Wert in einem Array-Objekt namens `oddNumbers`. Mit einer `trace()`-Anweisung wird jeder Wert im Format „oddNumber[*Index*] = *Wert*“ gedruckt:

```
var oddNumbers:Array = [1, 3, 5, 7, 9, 11];
var len:uint = oddNumbers.length;
for (var i:uint = 0; i < len; i++)
{
    trace("oddNumbers[" + i.toString() + "] = " + oddNumbers[i].toString());
}
```

Die Array-Klasse

Der erste Typ eines indizierten Arrays ist die Array-Klasse. Eine Array-Instanz kann einen Wert eines jeden Datentyps enthalten. Ein Array-Objekt kann Objekte enthalten, die unterschiedliche Datentypen aufweisen. So kann eine Array-Instanz beispielsweise bei Index 0 einen Stringwert enthalten, bei Index 1 eine Number-Instanz und bei Index 2 ein XML-Objekt.

Die Vector-Klasse

In ActionScript 3.0 ist als weiterer Typ eines indizierten Arrays die Vector-Klasse verfügbar. Bei einer Vector-Instanz handelt es sich um ein *Typ-Array*. Dies bedeutet, dass alle Elemente einer Vector-Instanz stets denselben Datentyp aufweisen.

Hinweis: *Die Vector-Klasse ist verfügbar ab Flash Player 10 und Adobe AIR 1.5.*

Beim Deklarieren einer Vector-Variablen oder beim Instanziieren eines Vector-Objekts geben Sie den Datentyp der enthaltenen Objekte explizit an. Der angegebene Datentyp wird als *Basistyp* des Vektors bezeichnet. Zur Laufzeit und bei der Kompilierung (im strikten Modus) wird der ganze Code überprüft, der den Wert eines Vector-Elements festlegt oder abruft. Wenn der Datentyp des hinzugefügten oder abgerufenen Objekts nicht mit dem Vector-Basistyp übereinstimmt, tritt ein Fehler auf.

Neben der Datentypbeschränkung unterscheidet sich die Vector-Klasse von der Array-Klasse noch durch weitere Beschränkungen:

- Ein Vector ist ein dichtes Array. Ein Array-Objekt kann Werte an den Indexpositionen 0 und 7 enthalten, selbst wenn die Indexpositionen 1 bis 6 unbelegt sind. Dagegen muss ein Vektor an jeder Indexposition einen Wert (oder `null`) aufweisen.
- Für einen Vektor kann wahlweise eine feste Länge angegeben werden. Dies bedeutet, dass sich die Anzahl der im Vektor enthaltenen Elemente nicht ändern kann.
- Der Zugriff auf die Vector-Elemente ist begrenzt. Es kann kein Wert von einer Indexposition größer als das letzte Element (`length` - 1) gelesen werden. Es kann kein Wert festgelegt werden, der eine Indexposition von mehr als eins hinter der aktuellen letzten Indexposition aufweist (anders ausgedrückt, ein Wert kann nur an einer vorhandenen Indexposition oder an der Indexposition `[length]` festgelegt werden).

Aufgrund seiner Beschränkungen bietet ein Vektor drei wichtige Vorteile gegenüber einer Array-Instanz, bei deren Elementen es sich ausschließlich um Instanzen einer Klasse handelt:

- Leistung: Bei einer Vector-Instanz erfolgen der Zugriff auf die Array-Elemente und die Iteration viel schneller als bei einer Array-Instanz.
- Typsicherheit: Im strikten Modus kann der Compiler Datentypfehler erkennen. Solche Fehler treten beispielsweise auf, wenn einem Vektor ein Wert mit einem falschen Datentyp zugewiesen wird oder wenn beim Lesen eines Wertes von einem Vektor der falsche Datentyp erwartet wird. Zur Laufzeit werden die Datentypen auch überprüft, wenn Daten einem Vector-Objekt hinzugefügt oder von einem Vector-Objekt gelesen werden. Beachten Sie jedoch Folgendes: Wenn Sie einem Vektor mithilfe der `push()`- oder `unshift()`-Methode Werte hinzufügen, werden die Datentypen der Argumente bei der Kompilierung nicht überprüft. Bei Verwendung dieser Methoden werden die Werte jedoch zur Laufzeit überprüft.
- Zuverlässigkeit: Die Laufzeit-Bereichsüberprüfung (oder Festlängenüberprüfung) erhöht die Zuverlässigkeit im Vergleich mit Arrays erheblich.

Abgesehen von den zusätzlichen Einschränkungen und Vorteilen entspricht die Vector-Klasse im Großen und Ganzen der Array-Klasse. Die Eigenschaften und Methoden eines Vector-Objekts ähneln den Eigenschaften und Methoden eines Arrays, in den meisten Fällen sind sie sogar identisch. In den meisten Fällen, in denen Sie ein Array verwenden würden, dessen Elemente alle denselben Datentyp aufweisen, ist es empfehlenswert, stattdessen eine Vector-Instanz zu verwenden.

Erstellen von Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Array- oder Vector-Instanzen können mit mehreren Verfahren erstellt werden. Die Techniken zur Erstellung der einzelnen Array-Typen unterscheiden sich jedoch etwas.

Erstellen einer Array-Instanz

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie erstellen ein Array-Objekt, indem Sie den `Array()`-Konstruktor aufrufen oder die Array-Literalsyntax verwenden.

Die `Array`-Konstruktorfunktion kann auf drei Arten verwendet werden. Wenn Sie erstens den Konstruktor ohne Argumente aufrufen, wird ein leeres Array zurückgegeben. Mithilfe der `length`-Eigenschaft der Array-Klasse können Sie überprüfen, ob das Array über keine Elemente verfügt. Im folgenden Code wird der `Array()`-Konstruktor beispielsweise ohne Argumente aufgerufen:

```
var names:Array = new Array();
trace(names.length); // output: 0
```

Wenn Sie zweitens eine Zahl als einzigen Parameter des `Array()`-Konstruktors verwenden, wird ein Array mit der angegebenen Länge erstellt. Die Werte aller Elemente sind dabei jeweils auf `undefined` gesetzt. Das Argument muss eine vorzeichenlose Ganzzahl zwischen 0 und 4.294.967.295 sein. Mit dem folgenden Code wird der `Array()`-Konstruktor beispielsweise mit einem numerischen Argument aufgerufen:

```
var names:Array = new Array(3);
trace(names.length); // output: 3
trace(names[0]); // output: undefined
trace(names[1]); // output: undefined
trace(names[2]); // output: undefined
```

Wenn Sie drittens den Konstruktor aufrufen und eine Liste mit Elementen als Parameter übergeben, wird ein Array mit Elementen erstellt, die jeweils den einzelnen Parametern entsprechen. Mit dem folgenden Code werden drei Argumente an den `Array()`-Konstruktor übergeben:

```
var names:Array = new Array("John", "Jane", "David");
trace(names.length); // output: 3
trace(names[0]); // output: John
trace(names[1]); // output: Jane
trace(names[2]); // output: David
```

Sie können auch Arrays mit Array-Literalen erstellen. Ein Array-Literal kann einer Array-Variablen direkt zugewiesen werden, wie im folgenden Beispiel gezeigt:

```
var names:Array = ["John", "Jane", "David"];
```

Erstellen einer Vector-Instanz

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Eine Vector-Instanz wird durch Aufrufen des `Vector.<T>()`-Konstruktors erstellt. Sie können einen Vektor auch erstellen, indem Sie die globale Funktion `Vector.<T>()` aufrufen. Diese Funktion wandelt ein angegebenes Objekt in eine Vector-Instanz um. In Flash Professional CS5 und höher, Flash Builder 4 und höher sowie in Flex 4 und höher können Sie eine Vector-Instanz auch erstellen, indem Sie eine Vector-Literalsyntax verwenden.

Wenn Sie eine Vector-Variable deklarieren (oder einen Vector-Methodenparameter oder den Rückgabetyp einer Methode), geben Sie immer den Basistyp der Vector-Variablen an. Sie geben den Basistyp auch an, wenn Sie eine Vector-Instanz erstellen, indem Sie den `Vector.<T>()`-Konstruktors erstellt. Anders ausgedrückt: Jede Verwendung des Begriffs `Vector` in ActionScript wird von einem Basistyp begleitet.

Sie geben den Vector-Basistyp mithilfe der type-Parametersyntax an. Der type-Parameter folgt im Code direkt auf das Wort `Vector`. Er besteht aus einem Punkt `.`) gefolgt vom Basisklassennamen in spitzen Klammern (`<>`), wie im folgenden Beispiel gezeigt:

```
var v:Vector.<String>;
v = new Vector.<String>();
```

In der ersten Zeile des Beispiels wird die Variable `v` als eine `Vector.<String>`-Instanz deklariert. Sie repräsentiert also ein indiziertes Array, das nur String-Instanzen enthalten kann. Mit der zweiten Zeile wird der `Vector()`-Konstruktor aufgerufen, um eine Instanz desselben Vector-Typs zu erstellen (also einen Vektor, dessen Elemente alle String-Objekte sind). Dieses Objekt wird `v` zugewiesen.

Verwenden des Vector.<T>()-Konstruktors

Wenn Sie den `Vector.<T>()`-Konstruktor ohne Argumente verwenden, wird eine leere Vector-Instanz erstellt. Sie können testen, ob der Vektor leer ist, indem Sie seine `length`-Eigenschaft überprüfen. Mit dem folgenden Beispielcode wird der `Vector.<T>()`-Konstruktor ohne Argumente aufgerufen:

```
var names:Vector.<String> = new Vector.<String>();
trace(names.length); // output: 0
```

Wenn bereits im Voraus bekannt ist, wie viele Elemente ein Vektor anfänglich benötigt, können Sie die Anzahl der Elemente im Vektor vorab festlegen. Zum Erstellen eines Vektors mit einer bestimmten Elementanzahl übergeben Sie die Anzahl der Elemente als ersten Parameter (`length`-Parameter). Da Vector-Elemente nicht leer sein dürfen, werden die Elemente mit Instanzen des Basistyps gefüllt. Wenn es sich beim Basistyp um einen Verweistyp handelt, der `null`-Werte zulässt, enthalten alle Elemente `null`. Andernfalls enthalten alle Elemente den Standardwert der Klasse. Beispielsweise kann eine uint-Variable nicht `null` sein. Deshalb wird im folgenden Codebeispiel der Vektor namens `ages` mit sieben Elementen erstellt, die alle den Wert 0 enthalten:

```
var ages:Vector.<uint> = new Vector.<uint>(7);
trace(ages); // output: 0,0,0,0,0,0,0
```

Schließlich können Sie mit dem `Vector.<T>()`-Konstruktor auch einen Vektor mit fester Länge erstellen. Dazu übergeben Sie den Wert `true` für den zweiten Parameter (`fixed`-Parameter). In diesem Fall wird der Vektor mit der angegebenen Elementanzahl erstellt, die sich nicht ändern kann. Die Werte für die Elemente eines Vektors mit fester Länge können jedoch nach wie vor geändert werden.

Verwenden des Vector-Literalsyntaxkonstruktors

In Flash Professional CS5 und höher, Flash Builder 4 und höher sowie Flex 4 und höher können Sie eine Liste mit Werten an den `Vector.<T>()`-Konstruktor übergeben, um die anfänglichen Vector-Werte festzulegen:

```
// var v:Vector.<T> = new <T>[E0, ..., En-1 ,];
// For example:
var v:Vector.<int> = new <int>[0,1,2,];
```

Für diese Syntax gelten die folgenden Informationen:

- Das nachgestellte Komma ist optional.
- Leere Elemente im Array werden nicht unterstützt; eine Anweisung wie `var v:Vector.<int> = new <int>[0,,2,]` führt zu einem Compilerfehler.

- Sie können keine Standardlänge für die Vektorinstanz festlegen. Stattdessen ist die Länge mit der Anzahl der Elemente in der Initialisierungsliste identisch.
- Sie können nicht angeben, ob die Vector-Instanz eine feste Länge hat. Verwenden Sie stattdessen die `fixed`-Eigenschaft.
- Datenverluste oder -fehler können auftreten, wenn Elemente, die als Werte übergeben wurden, nicht dem angegebenen Typ entsprechen. Zum Beispiel:

```
var v:Vector.<int> = new <int>[4.2]; // compiler error when running in strict mode
trace(v[0]); //returns 4 when not running in strict mode
```

Verwenden des Vector.<T>()-Funktion

Zusätzlich zum `Vector.<T>()`- und zum Vector-Literalsyntaxkonstruktor können Sie auch die globale `Vector.<T>()`-Funktion verwenden, um ein Vector-Objekt zu erstellen. Bei der globalen `Vector.<T>()`-Funktion handelt es sich um eine Umwandlungsfunktion. Beim Aufruf der globalen `Vector.<T>()`-Funktion geben Sie den Vector-Basistyp an, der von der Methode zurückgegeben wird. Sie übergeben ein einzelnes indiziertes Array (Array- oder Vector-Instanz) als Argument. Die Methode gibt dann einen Vektor mit dem angegebenen Basistyp zurück, der die Werte im Argument des Quell-Arrays enthält. Das folgende Codebeispiel zeigt die Syntax für einen Aufruf der globalen `Vector.<T>()`-Funktion:

```
var friends:Vector.<String> = Vector.<String>(["Bob", "Larry", "Sarah"]);
```

Bei der globalen `Vector.<T>()`-Funktion wird eine Datenumwandlung auf zwei Ebenen durchgeführt. Erstens wird bei der Übergabe einer Array-Instanz an die Funktion eine Vector-Instanz zurückgegeben. Zweitens versucht die Funktion, die Elemente des Quell-Arrays in Werte des Basistyps umzuwandeln, unabhängig davon, ob es sich beim Quell-Array um eine Array- oder eine Vector-Instanz handelt. Bei der Umwandlung gelten die ActionScript-Standardregeln für die Datentypumwandlung. Im folgenden Codebeispiel werden die String-Werte im Quell-Array in Ganzzahlen im Ergebnisvektor umgewandelt. Im Ergebnis wird der Dezimalteil des ersten Wertes („`1.5`") gekürzt und der nicht numerische dritte Wert („`Waffles`") wird in 0 umgewandelt:

```
var numbers:Vector.<int> = Vector.<int>(["1.5", "17", "Waffles"]);
trace(numbers); // output: 1,17,0
```

Wenn eines der Quellelemente nicht umgewandelt werden kann, tritt ein Fehler auf.

Wenn beim Aufruf der globalen `Vector.<T>()`-Funktion ein Element im Quell-Array eine Instanz einer Unterklasse des angegebenen Basistyps ist, wird das Element dem Ergebnisvektor hinzugefügt. In diesem Fall tritt kein Fehler auf. Die globale Vector.`<T>()`-Funktion bietet die einzige Möglichkeit, um einen Vektor mit dem Basistyp `T` in einen Vektor umzuwandeln, bei dessen Basistyp es sich um eine übergeordnete Klasse von `T` handelt.

Einfügen von Array-Elementen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Am einfachsten können Sie einem indizierten Array ein Element hinzufügen, indem Sie den Array-Zugriffsoperator (`[]`) verwenden. Um den Wert eines Elements in einem indizierten Array festzulegen, verwenden Sie den Namen des Array- oder Vector-Objekts und die Indexnummer auf der linken Seite einer Zuweisungsanweisung:

```
songTitles[5] = "Happy Birthday";
```

Wenn im Array- oder Vector-Objekt noch kein Element an dieser Indexposition vorhanden ist, wird die Indexposition erstellt und der Wert wird an dieser Position gespeichert. Ist bereits ein Wert an dieser Indexposition vorhanden, wird der vorhandene Wert durch den neuen Wert ersetzt.

Mithilfe eines Array-Objekts können Sie an jeder Indexposition ein Element erstellen. Dagegen ist es mit einem Vector-Objekt nur möglich, eine vorhandene Indexposition oder der nächsten verfügbaren Indexposition einen Wert zuzuweisen. Die nächste verfügbare Indexposition entspricht der `length`-Eigenschaft des Vector-Objekts. Die sicherste Methode, einem Vector-Objekt ein neues Element hinzuzufügen, wird im folgenden Codebeispiel veranschaulicht:

```
myVector[myVector.length] = valueToAdd;
```

Mithilfe von drei Methoden der Array- und Vector-Klassen – `push()`, `unshift()` und `splice()` – können Sie Elemente in ein indiziertes Array einfügen. Mit der `push()`-Methode werden ein oder mehrere Elemente am Ende eines Arrays angehängt. Mit anderen Worten, das letzte mithilfe der `push()`-Methode in das Array eingefügte Element weist die höchste Indexnummer auf. Mit der `unshift()`-Methode werden ein oder mehrere Elemente am Anfang eines Arrays eingefügt, und zwar immer bei Indexposition 0. Mit der `splice()`-Methode wird eine beliebige Anzahl von Elementen an einer bestimmten Indexposition in einem Array eingefügt.

Im folgenden Beispiel werden alle drei Methoden veranschaulicht. Ein Array namens `planets` wird erstellt, um die Namen der Planeten in der Reihenfolge ihrer Nähe zur Sonne zu speichern. Als Erstes wird die `push()`-Methode aufgerufen, um das erste Element, `Mars`, hinzuzufügen. Dann wird die `unshift()`-Methode aufgerufen, um das Element `Mercury` (Merkur) am Anfang des Arrays einzufügen. Schließlich wird die `splice()`-Methode aufgerufen, um die Elemente `Venus` und `Earth` (Erde) nach `Mercury`, jedoch vor `Mars` einzufügen. Mit dem ersten an `splice()` übergebenen Argument, der Ganzzahl 1, wird angegeben, dass die Elemente an der Indexposition 1 eingefügt werden sollen. Mit dem zweiten an `splice()` gesendeten Argument, der Ganzzahl 0, wird angegeben, dass keine Elemente gelöscht werden sollen. Bei dem dritten und vierten Argument, `Venus` und `Earth`, die an `splice()` gesendet werden, handelt es sich um die einzufügenden Elemente.

```
var planets:Array = new Array();
planets.push("Mars"); // array contents: Mars
planets.unshift("Mercury"); // array contents: Mercury,Mars
planets.splice(1, 0, "Venus", "Earth");
trace(planets); // array contents: Mercury,Venus,Earth,Mars
```

Mit den Methoden `push()` und `unshift()` wird jeweils eine vorzeichenlose Ganzzahl zurückgegeben, die die Länge des geänderten Arrays angibt. Wenn die `splice()`-Methode zum Einfügen von Elementen verwendet wird, wird ein leeres Array zurückgegeben. Dies ist auf die Vielseitigkeit der `splice()`-Methode zurückzuführen. Sie können die `splice()`-Methode nicht nur zum Einfügen von Elementen in ein Array, sondern auch zum Entfernen von Elementen aus einem Array verwenden. Wenn die `splice()`-Methode zum Entfernen von Elementen verwendet wird, wird ein Array mit den entfernten Elementen zurückgegeben.

Hinweis: *Wenn die `fixed`-Eigenschaft eines Vector-Objekts auf `true` eingestellt ist, kann die Gesamtanzahl der Elemente im Vektor sich nicht ändern. Wenn Sie versuchen, einem Vector-Objekt mit fester Länge mithilfe der hier beschriebenen Verfahren ein neues Element hinzuzufügen, tritt ein Fehler auf.*

Abrufen von Werten und Entfernen von Array-Elementen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Am einfachsten können Sie den Wert eines Elements aus einem indizierten Array abrufen, indem Sie den Array-Zugriffsoperator (`[]`) verwenden. Um den Wert eines Elements in einem indizierten Array abzurufen, verwenden Sie den Namen des Array- oder Vector-Objekts und die Indexnummer auf der rechten Seite einer Zuweisungsanweisung:

```
var myFavoriteSong:String = songTitles[3];
```

Es kann versucht werden, den Wert von einem Array- oder Vector-Objekt mithilfe eines Indexes abzurufen, an dem keine Elemente existieren. In diesem Fall gibt ein Array-Objekt den Wert „undefined“ zurück und ein Vector-Objekt gibt eine RangeError-Ausnahme aus.

Mit drei Methoden der Array- und Vector-Klassen – `pop()`, `shift()` und `splice()` – können Sie Elemente entfernen. Mit der `pop()`-Methode wird ein Element am Ende eines Arrays entfernt. Dies bedeutet, dass das Element mit der höchsten Indexnummer entfernt wird. Mit der `shift()`-Methode wird ein Element am Anfang eines Arrays entfernt. Dies bedeutet, dass immer das Element an Indexposition 0 entfernt wird. Mit der `splice()`-Methode, die auch zum Einfügen von Elementen verwendet werden kann, wird eine beliebige Anzahl von Elementen entfernt, angefangen bei der Indexposition, die durch das erste an die Methode gesendete Argument angegeben ist.

Im folgenden Beispiel werden Elemente mithilfe aller drei Methoden aus einer Array-Instanz entfernt. Ein Array namens `oceans` (Ozeane) wird erstellt. In diesem Array sollen die Namen großer Gewässer gespeichert werden. Einige der Elemente im Array sind Seen und keine Meere und sollen daher entfernt werden.

Mithilfe der `splice()`-Methode werden zunächst die Elemente `Aral` und `Superior` entfernt sowie die Elemente `Atlantic` und `Indian` eingefügt. Mit dem ersten an `splice()` gesendeten Argument, der Ganzzahl 2, wird angegeben, dass der Vorgang beim dritten Element in der Liste starten soll, an Indexposition 2. Mit dem zweiten Argument, 2, wird angegeben, dass zwei Elemente entfernt werden sollen. Die restlichen Argumente, `Atlantic` und `Indian`, sind die Werte, die bei Indexposition 2 eingefügt werden sollen.

Mithilfe der `pop()`-Methode wird dann das letzte Element, `Huron`, im Array entfernt. Schließlich wird mithilfe der `shift()`-Methode das erste Element, `Victoria`, im Array entfernt.

```
var oceans:Array = ["Victoria", "Pacific", "Aral", "Superior", "Indian", "Huron"];
oceans.splice(2, 2, "Arctic", "Atlantic"); // replaces Aral and Superior
oceans.pop(); // removes Huron
oceans.shift(); // removes Victoria
trace(oceans);// output: Pacific,Arctic,Atlantic,Indian
```

Bei den Methoden `pop()` und `shift()` werden jeweils die entfernten Elemente zurückgegeben. Bei einer Array-Instanz hat der Rückgabewert den Object-Datentyp, da Arrays Werte eines beliebigen Datentyps enthalten können. Bei einer Vector-Instanz entspricht der Datentyp des Rückgabewertes dem Vector-Basistyp. Die `splice()`-Methode gibt ein Array oder einen Vektor mit den Werten zurück, die entfernt wurden. Sie können das `oceans`-Array so ändern, dass das zurückgegebene Array durch Aufrufen von `splice()` einer neuen Array-Variablen zugewiesen wird, wie im folgenden Beispiel gezeigt:

```
var lakes:Array = oceans.splice(2, 2, "Arctic", "Atlantic");
trace(lakes); // output: Aral,Superior
```

Möglicherweise stoßen Sie auf Code, bei dem der `delete`-Operator für ein Array-Objektelement verwendet wird. Mit dem `delete`-Operator wird der Wert eines Array-Elements auf `undefined` gesetzt, das Element wird jedoch nicht aus dem Array entfernt. Im folgenden Codebeispiel wird der `delete`-Operator für das dritte Element des `oceans`-Arrays verwendet, die Länge des Arrays bleibt jedoch unverändert bei 5:

```
var oceans:Array = ["Arctic", "Pacific", "Victoria", "Indian", "Atlantic"];
delete oceans[2];
trace(oceans);// output: Arctic,Pacific,,Indian,Atlantic
trace(oceans[2]); // output: undefined
trace(oceans.length); // output: 5
```

Sie können ein Array oder einen Vektor mithilfe der `length`-Eigenschaft kürzen. Wenn Sie die `length`-Eigenschaft eines indizierten Arrays auf eine geringere Länge als die derzeitige Array-Länge einstellen, wird das Array gekürzt. Dabei werden die Elemente an den Indexpositionen entfernt, die den neuen Wert für `length` minus 1 übersteigen. Wenn das `oceans`-Array beispielsweise so sortiert ist, dass alle gültigen Einträge sich am Anfang des Arrays befinden, können Sie die `length`-Eigenschaft verwenden, um alle Einträge am Ende des Arrays zu entfernen, wie im folgenden Codebeispiel gezeigt:

```
var oceans:Array = ["Arctic", "Pacific", "Victoria", "Aral", "Superior"];
oceans.length = 2;
trace(oceans); // output: Arctic,Pacific
```

Hinweis: *Wenn die `fixed`-Eigenschaft eines Vector-Objekts auf `true` eingestellt ist, kann die Gesamtanzahl der Elemente im Vektor sich nicht ändern. Wenn Sie versuchen, mit den hier beschriebenen Verfahren einen Vektor mit fester Länge zu kürzen oder ein Element aus einem solchen Vektor zu entfernen, tritt ein Fehler auf.*

Sortieren von Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe von drei Methoden – `reverse()`, `sort()` und `sortOn()` – können Sie die Reihenfolge in einem indizierten Array ändern, indem Sie die Elemente sortieren oder die bestehende Reihenfolge umkehren. Mit allen diesen Methoden wird ein vorhandenes Array geändert. In der folgenden Tabelle werden diese Methoden sowie ihr Verhalten bei Array- und Vector-Objekten zusammengefasst:

Methode	Array-Verhalten	Vector-Verhalten
`reverse()`	Ändert die Reihenfolge der Elemente so, dass das letzte Element zum ersten Element wird, das vorletzte Element zum zweiten Element und so weiter.	Mit Array-Verhalten identisch
`sort()`	Ermöglicht Ihnen das Sortieren der Array-Elemente in mehreren vordefinierten Reihenfolgen, wie beispielsweise alphabetisch oder numerisch. Sie können auch einen benutzerdefinierten Sortieralgorithmus angeben.	Sortiert die Elemente nach dem von Ihnen angegebenen benutzerdefinierten Sortieralgorithmus
`sortOn()`	Ermöglicht Ihnen das Sortieren von Objekten, die eine oder mehrere gemeinsame Eigenschaften aufweisen, wobei Sie die Eigenschaften angeben, die als Sortierschlüssel verwendet werden sollen.	In der Vector-Klasse nicht verfügbar

Die reverse()-Methode

Bei der `reverse()`-Methode werden keine Parameter verwendet und keine Werte zurückgegeben, Sie können mit dieser Methode jedoch die aktuelle Reihenfolge der Elemente in einem Array in die umgekehrte Reihenfolge ändern. Im folgenden Beispiel wird die Reihenfolge der im `oceans`-Array aufgeführten Meere geändert:

```
var oceans:Array = ["Arctic", "Atlantic", "Indian", "Pacific"];
oceans.reverse();
trace(oceans); // output: Pacific,Indian,Atlantic,Arctic
```

Einfaches Sortieren mit der sort()-Methode (nur Array-Klasse)

Bei einer Array-Instanz ändert die `sort()`-Methode die Anordnung der Elemente in einem Array entsprechend der *Standardsortierreihenfolge*. Die Standardsortierreihenfolge weist die folgenden Merkmale auf:

- Bei der Sortierung wird die Groß- und Kleinschreibung beachtet. Dabei stehen Großbuchstaben vor Kleinbuchstaben. Der Buchstabe D steht beispielsweise vor dem Buchstaben b.
- Die Sortierung erfolgt aufsteigend, d. h. niedrige Buchstabencodes (z. B. A) sind höheren Buchstabencodes (z. B. B) vorangestellt.
- Bei der Sortierung werden identische Werte nebeneinander, jedoch in keiner bestimmten Reihenfolge positioniert.
- Die Sortierung ist stringbasiert, d. h. Elemente werden vor dem Vergleichen in Strings konvertiert (10 steht z. B. vor 3, da der String „`1`" einen niedrigeren Buchstabencode hat als der String „`3`").

Möglicherweise möchten Sie ein Array ohne Berücksichtigung der Groß- und Kleinschreibung oder in absteigender Reihenfolge sortieren. Unter Umständen enthält ein Array zudem Zahlen, die Sie numerisch und nicht alphabetisch sortieren möchten. Die `sort()`-Methode der Array-Klasse verfügt über einen `options`-Parameter, über den Sie die einzelnen Merkmale der Standardsortierreihenfolge ändern können. In der folgenden Aufstellung sind die einzelnen Optionen aufgeführt, die durch mehrere statische Konstanten der Array-Klasse definiert sind:

- `Array.CASEINSENSITIVE`: Bei Angabe diese Option wird die Groß- und Kleinschreibung bei der Sortierung nicht berücksichtigt. Der Kleinbuchstabe b steht dann beispielsweise vor dem Großbuchstaben D.
- `Array.DESCENDING`: Hierdurch wird die aufsteigende Standardsortierreihenfolge umgekehrt. Der Buchstabe B steht dann beispielsweise vor dem Buchstaben A.
- `Array.UNIQUESORT`: Hierdurch wird die Sortierung bei zwei identischen Werten abgebrochen.
- `Array.NUMERIC`: Hierdurch erfolgt eine numerische Sortierung, sodass 3 vor 10 steht.

Im folgenden Beispiel werden einige dieser Optionen aufgegriffen. Das Array namens `poets` (Dichter) wird erstellt und anhand mehrerer Sortieroptionen sortiert.

```
var poets:Array = ["Blake", "cummings", "Angelou", "Dante"];
poets.sort(); // default sort
trace(poets); // output: Angelou,Blake,Dante,cummings

poets.sort(Array.CASEINSENSITIVE);
trace(poets); // output: Angelou,Blake,cummings,Dante

poets.sort(Array.DESCENDING);
trace(poets); // output: cummings,Dante,Blake,Angelou

poets.sort(Array.DESCENDING | Array.CASEINSENSITIVE); // use two options
trace(poets); // output: Dante,cummings,Blake,Angelou
```

Benutzerdefiniertes Sortieren mit der sort()-Methode (Array- und Vector-Klassen)

Sie können nicht nur die einfachen Sortierungen verwenden, die für ein Array-Objekt verfügbar sind, sondern zudem auch eine benutzerdefinierte Sortierregel erstellen. Für die Vector-Klasse steht ausschließlich diese Technik der `sort()`-Methode zur Verfügung. Zum Definieren einer benutzerdefinierten Sortierung erstellen Sie eine benutzerdefinierte Sortierfunktion, die Sie als Argument an die `sort()`-Methode übergeben.

Wenn Sie eine Liste mit Namen, bei der alle Elemente den vollständigen Namen einer Person enthalten, beispielsweise nach Nachnamen sortieren möchten, müssen Sie eine benutzerdefinierte Sortierfunktion verwenden, damit alle Elemente analysiert werden und bei der Sortierfunktion der Nachname verwendet wird. Im folgenden Codebeispiel ist eine benutzerdefinierte Funktion dargestellt, die als Parameter der `Array.sort()`-Methode verwendet wird:

```
var names:Array = new Array("John Q. Smith", "Jane Doe", "Mike Jones");
function orderLastName(a, b):int
{
    var lastName:RegExp = /\b\S+$/;
    var name1 = a.match(lastName);
    var name2 = b.match(lastName);
    if (name1 < name2)
    {
        return -1;
    }
    else if (name1 > name2)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
trace(names); // output: John Q. Smith,Jane Doe,Mike Jones
names.sort(orderLastName);
trace(names); // output: Jane Doe,Mike Jones,John Q. Smith
```

Bei der benutzerdefinierten Sortierfunktion `orderLastName()` wird mithilfe eines regulären Ausdrucks der Nachname jedes Elements extrahiert und für den Vergleichsvorgang verwendet. Der Funktionsbezeichner `orderLastName` wird als einziger Parameter beim Aufrufen der `sort()`-Methode für das `names`-Array verwendet. Für die Sortierfunktion werden zwei Parameter (`a` und `b`) angegeben, da die Funktion auf zwei Array-Elemente gleichzeitig angewendet wird. Der Rückgabewert der Sortierfunktion gibt die Sortierreihenfolge der Elemente an.

- Der Rückgabewert -1 gibt an, dass der erste Parameter `a` vor dem zweiten Parameter `b` steht.
- Der Rückgabewert 1 gibt an, dass der zweite Parameter `b` vor dem ersten Parameter `a` steht.
- Der Rückgabewert 0 gibt an, dass die Sortierrangfolge der Elemente gleichwertig ist.

Die sortOn()-Methode (nur Array-Klasse)

Die `sortOn()`-Methode wird bei Array-Objekten mit Elementen eingesetzt, die Objekte enthalten. Diese Objekte müssen über mindestens eine gemeinsame Eigenschaft verfügen, die als Sortierschlüssel verwendet werden kann. Die Verwendung der `sortOn()`-Methode bei Arrays mit anderen Datentypen führt zu unerwarteten Ergebnissen.

Hinweis: *Die Vector-Klasse enthält keine `sortOn()`-Methode. Diese Methode ist nur für Array-Objekte verfügbar.*

Im folgenden Beispiel wird das `poets`-Array so geändert, dass jedes Element ein Objekt und kein String ist. Jedes Objekt enthält den Nachnamen und das Geburtsjahr eines Dichters.

```
var poets:Array = new Array();
poets.push({name:"Angelou", born:"1928"});
poets.push({name:"Blake", born:"1757"});
poets.push({name:"cummings", born:"1894"});
poets.push({name:"Dante", born:"1265"});
poets.push({name:"Wang", born:"701"});
```

Mithilfe der `sortOn()`-Methode können Sie das Array nach der `born`-Eigenschaft sortieren. Mit der `sortOn()`-Methode werden die beiden Parameter `fieldName` und `options` definiert. Das `fieldName`-Argument muss als String angegeben werden. Im folgenden Beispiel wird `sortOn()` mit den beiden Argumenten `born` und `Array.NUMERIC` aufgerufen. Durch das `Array.NUMERIC`-Argument wird sichergestellt, dass die Sortierung numerisch und nicht alphabetisch durchgeführt wird. Dies empfiehlt sich auch, wenn die Zahlen die gleiche Anzahl an Ziffern aufweisen, da dadurch ein normaler Sortiervorgang durchgeführt werden kann, falls zu einem späteren Zeitpunkt eine Zahl mit mehr oder weniger Ziffern in das Array eingefügt wird.

```
poets.sortOn("born", Array.NUMERIC);
for (var i:int = 0; i < poets.length; ++i)
{
    trace(poets[i].name, poets[i].born);
}
/* output:
Wang 701
Dante 1265
Blake 1757
cummings 1894
Angelou 1928
*/
```

Sortieren ohne Änderung des ursprünglichen Arrays (nur Array-Klasse)

Im Allgemeinen werden mit den Methoden `sort()` und `sortOn()` Änderungen an einem Array vorgenommen. Wenn Sie ein Array sortieren möchten, ohne Änderungen vorzunehmen, geben Sie die Konstante `Array.RETURNINDEXEDARRAY` als Teil des `options`-Parameters an. Dadurch wird ein neues Array mit der gewünschten Sortierung zurückgegeben und das ursprüngliche Array wird nicht geändert. Das zurückgegebene Array ist ein einfaches Array mit Indexnummern, die die neue Sortierreihenfolge angeben, und enthält keine Elemente des ursprünglichen Arrays. Wenn Sie das `poets`-Array ohne Änderungen nach Geburtsjahr sortieren möchten, geben Sie die Konstante `Array.RETURNINDEXEDARRAY` als Bestandteil des an den `options`-Parameter übergebenen Arguments an.

Im folgenden Beispiel werden die zurückgegebenen Indexinformationen in einem Array namens `indices` gespeichert. Dann werden über das `indices`-Array in Verbindung mit dem nicht geänderten `poets`-Array die Dichter nach dem entsprechenden Geburtsjahr sortiert ausgegeben:

```
var indices:Array;
indices = poets.sortOn("born", Array.NUMERIC | Array.RETURNINDEXEDARRAY);
for (var i:int = 0; i < indices.length; ++i)
{
    var index:int = indices[i];
    trace(poets[index].name, poets[index].born);
}
/* output:
Wang 701
Dante 1265
Blake 1757
cummings 1894
Angelou 1928
*/
```

Abfragen von Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit vier Methoden der Array- und Vector-Klassen – `concat()`, `join()`, `slice()` und `toString()` – werden Daten im Array abgefragt, jedoch keine Änderungen am Array vorgenommen. Bei den Methoden `concat()` und `slice()` werden jeweils neue Arrays zurückgegeben, während bei den Methoden `join()` und `toString()` jeweils Strings zurückgegeben werden. Bei der `concat()`-Methode wird als Argument ein neues Array oder eine Liste mit Elementen mit dem vorhandenen Array verbunden, um ein neues Array zu erstellen. Die `slice()`-Methode verfügt über zwei Parameter mit den treffenden Bezeichnungen `startIndex` und `endIndex` und gibt ein neues Array zurück, das eine Kopie eines Teils der Elemente aus dem vorhandenen Array enthält. Der Teil beginnt mit dem Element bei `startIndex` und endet mit dem Element vor `endIndex`. Noch einmal zur Klarstellung: Das Element bei `endIndex` ist nicht Bestandteil des Rückgabewerts.

Im folgenden Beispiel werden mithilfe von `concat()` und `slice()` neue Arrays mit Elementen aus anderen Arrays erstellt:

```
var array1:Array = ["alpha", "beta"];
var array2:Array = array1.concat("gamma", "delta");
trace(array2); // output: alpha,beta,gamma,delta

var array3:Array = array1.concat(array2);
trace(array3); // output: alpha,beta,alpha,beta,gamma,delta

var array4:Array = array3.slice(2,5);
trace(array4); // output: alpha,beta,gamma
```

Mithilfe der Methoden `join()` und `toString()` können Sie ein Array abfragen und den entsprechenden Inhalt in einem String zurückgeben. Wenn bei der `join()`-Methode keine Parameter verwendet werden, sind beide Methoden identisch, d. h., sie geben einen String mit einer durch Kommas getrennten Liste aller Elemente im Array zurück. Im Gegensatz zur `toString()`-Methode kann bei der `join()`-Methode der `delimiter`-Parameter übergeben werden, über den Sie das Symbol auswählen können, das im zurückgegebenen String als Trennzeichen zwischen den einzelnen Elementen verwendet wird.

Im folgenden Beispiel wird ein Array namens `rivers` (Flüsse) erstellt, für das `join()` und `toString()` aufgerufen werden, um die Werte im Array als String zurückzugeben. Die `toString()`-Methode wird verwendet, damit durch Kommas getrennte Werte zurückgegeben werden (`riverCSV`). Die `join()`-Methode wird verwendet, damit Werte zurückgegeben werden, die durch `+`-Zeichen getrennt sind.

```
var rivers:Array = ["Nile", "Amazon", "Yangtze", "Mississippi"];
var riverCSV:String = rivers.toString();
trace(riverCSV); // output: Nile,Amazon,Yangtze,Mississippi
var riverPSV:String = rivers.join("+");
trace(riverPSV); // output: Nile+Amazon+Yangtze+Mississippi
```

Beachten Sie, dass verschachtelte Array- oder Vector-Instanzen bei der `join()`-Methode immer mit durch Kommas getrennten Werten zurückgegeben werden, unabhängig davon, welches Trennzeichen für die Hauptelemente im Array angegeben wird, wie im folgenden Beispiel gezeigt:

```
var nested:Array = ["b","c","d"];
var letters:Array = ["a",nested,"e"];
var joined:String = letters.join("+");
trace(joined); // output: a+b,c,d+e
```

Assoziative Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein assoziatives Array wird manchmal auch als *Hash* oder *Zuordnung* bezeichnet und organisiert gespeicherte Werte mithilfe von *Schlüsseln* anstatt mit einem numerischen Index. Jeder Schlüssel in einem assoziativen Array ist ein eindeutiger String, über den auf einen gespeicherten Wert zugegriffen wird. Ein assoziatives Array ist eine Instanz der Object-Klasse, d. h. jeder Schlüssel entspricht einem Eigenschaftsnamen. Assoziative Arrays sind ungeordnete Sammlungen aus Schlüssel- und Wert-Paaren. Im Code kann nicht vorausgesetzt werden, dass die Schlüssel eines assoziativen Arrays in einer bestimmten Reihenfolge aufgeführt sind.

ActionScript 3.0 bietet auch einen erweiterten Typ assoziativer Arrays mit der Bezeichnung *dictionary* (Wörterbuch). Wörterbucher sind Instanzen der Dictionary-Klasse im Paket „flash.utils“. Sie verwenden Schlüssel, die einen beliebigen Datentyp aufweisen können. Mit anderen Worten: Schlüssel für Wörterbücher sind nicht auf Werte mit dem String-Datentyp beschränkt.

Assoziative Arrays mit Stringschlüsseln

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Assoziative Arrays können in ActionScript 3.0 mit zwei verschiedenen Techniken erstellt werden. Bei der ersten Technik wird eine Object-Instanz verwendet. Bei Verwendung einer Object-Instanz können Sie Ihr Array mit einem Objektliteral initialisieren. Eine Instanz der Object-Klasse, auch als *generisches Objekt* bezeichnet, ist funktional mit einem assoziativen Array identisch. Jeder Eigenschaftsname des generischen Objekts dient als Schlüssel, der den Zugriff auf einen gespeicherten Wert ermöglicht.

Im folgenden Beispiel wird das assoziative Array `monitorInfo` erstellt und mithilfe eines Objektliterals mit zwei Schlüssel-Wert-Paaren initialisiert:

```
var monitorInfo:Object = {type:"Flat Panel", resolution:"1600 x 1200"};
trace(monitorInfo["type"], monitorInfo["resolution"]);
// output: Flat Panel 1600 x 1200
```

Wenn das Array nicht bei der Deklaration initialisiert werden muss, können Sie das Array wie folgt mit dem Object-Konstruktor erstellen:

```
var monitorInfo:Object = new Object();
```

Nachdem Sie das Array mit einem Objektliteral oder dem Konstruktor der Object-Klasse erstellt haben, können Sie ihm neue Werte hinzufügen, indem Sie entweder den Array-Zugriffsoperator (`[]`) oder den Punktoperator (`.`) verwenden. Im folgenden Beispiel werden `monitorArray` zwei neue Werte hinzugefügt:

```
monitorInfo["aspect ratio"] = "16:10"; // bad form, do not use spaces
monitorInfo.colors = "16.7 million";
trace(monitorInfo["aspect ratio"], monitorInfo.colors);
// output: 16:10 16.7 million
```

Beachten Sie, dass der Schlüssel `aspect ratio` ein Leerzeichen enthält. Dies ist bei Verwendung des Array-Zugriffsoperators (`[]`) möglich, führt jedoch bei Verwendung des Punktoperators zu einer Fehlermeldung. Es wird davon abgeraten, Leerzeichen in Schlüsselnamen zu verwenden.

Beim zweiten Verfahren zur Erstellung eines assoziativen Arrays wird der Array-Konstruktor (bzw. der Konstruktor einer beliebigen dynamischen Klasse) verwendet. Mit dem Array-Zugriffsoperator (`[]`) oder dem Punktoperator (`.`) werden anschließend Schlüssel-Wert-Paare zum Array hinzugefügt. Wenn Sie das assoziative Array als Array-Typ deklarieren, kann das Array nicht mit einem Objektliteral initialisiert werden. Im folgenden Beispiel werden mit dem Array-Konstruktor das assoziative Array `monitorInfo` erstellt sowie der Schlüssel `type` und der Schlüssel `resolution` mit den zugehörigen Werten hinzugefügt:

```
var monitorInfo:Array = new Array();
monitorInfo["type"] = "Flat Panel";
monitorInfo["resolution"] = "1600 x 1200";
trace(monitorInfo["type"], monitorInfo["resolution"]);
// output: Flat Panel 1600 x 1200
```

Die Verwendung des Array-Konstruktors zur Erstellung eines assoziativen Arrays bietet keine Vorteile. Bei assoziativen Arrays können Sie nicht die `Array.length`-Eigenschaft und keine Methoden der Array-Klasse verwenden, auch wenn Sie den Array-Konstruktor oder den Array-Datentyp verwenden. Die Verwendung des Array-Konstruktors eignet sich am besten zur Erstellung indizierter Arrays.

Assoziative Arrays mit Objektschlüsseln (Wörterbücher)

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der Dictionary-Klasse können Sie assoziative Arrays erstellen, bei denen Objekte und keine Strings als Schlüssel verwendet werden. Diese Arrays werden auch als Wörterbücher, Hashes oder Zuordnungen bezeichnet. Bei einer Anwendung, bei der die Position eines Sprite-Objekts beispielsweise anhand seiner Zuordnung zu einem bestimmten Container festgelegt werden soll, können Sie mithilfe eines Dictionary-Objekts jedes Sprite-Objekt einem Container zuordnen.

Mit dem folgenden Code werden drei Instanzen der Sprite-Klasse erstellt, die als Schlüssel für das Dictionary-Objekt dienen. Jedem Schlüssel wird als Wert entweder `GroupA` oder `GroupB` zugewiesen. Die Werte können jeden Datentyp aufweisen. In diesem Beispiel sind `GroupA` und `GroupB` jedoch Instanzen der Object-Klasse. Anschließend können Sie mit dem Array-Zugriffsoperator (`[]`) auf den Wert zugreifen, der mit den einzelnen Schlüsseln verknüpft ist, wie im folgenden Codebeispiel gezeigt:

```
import flash.display.Sprite;
import flash.utils.Dictionary;

var groupMap:Dictionary = new Dictionary();

// objects to use as keys
var spr1:Sprite = new Sprite();
var spr2:Sprite = new Sprite();
var spr3:Sprite = new Sprite();

// objects to use as values
var groupA:Object = new Object();
var groupB:Object = new Object();

// Create new key-value pairs in dictionary.
groupMap[spr1] = groupA;
groupMap[spr2] = groupB;
groupMap[spr3] = groupB;

if (groupMap[spr1] == groupA)
{
    trace("spr1 is in groupA");
}
if (groupMap[spr2] == groupB)
{
    trace("spr2 is in groupB");
}
if (groupMap[spr3] == groupB)
{
    trace("spr3 is in groupB");
}
```

Iteration mit Objektschlüsseln

Sie können eine Iteration für den Inhalt des Dictionary-Objekts mit einer `for..in`-Schleife oder einer `for each..in`-Schleife durchführen. Mit einer `for..in`-Schleife können Sie die Iteration basierend auf den Schlüsseln durchführen, während bei einer `for each..in`-Schleife die Iteration auf den jedem Schlüssel zugeordneten Werten basiert.

Verwenden Sie die `for..in`-Schleife für den direkten Zugriff auf die Objektschlüssel eines Dictionary-Objekts. Sie können auch über den Array-Zugriffsoperator (`[]`) auf die Werte des Dictionary-Objekts zugreifen. Im folgenden Codebeispiel wird das vorherige Beispiel des `groupMap`-Wörterbuchs verwendet, um die Iteration eines Dictionary-Objekts mit der `for..in`-Schleife durchzuführen:

```
for (var key:Object in groupMap)
{
    trace(key, groupMap[key]);
}
/* output:
[object Sprite] [object Object]
[object Sprite] [object Object]
[object Sprite] [object Object]
*/
```

Verwenden Sie die `for each..in`-Schleife für den direkten Zugriff auf die Werte eines Dictionary-Objekts. Im folgenden Codebeispiel wird erneut das `groupMap`-Wörterbuch verwendet, um die Iteration eines Dictionary-Objekts mit der `for each..in`-Schleife durchzuführen:

```
for each (var item:Object in groupMap)
{
    trace(item);
}
/* output:
[object Object]
[object Object]
[object Object]
*/
```

Objektschlüssel und Speicherverwaltung

In Adobe® Flash® Player und Adobe® AIR™ wird mit einem Verfahren zur automatischen Speicherbereinigung (auch Garbage Collection genannt) Speicherplatz freigegeben, der nicht mehr genutzt wird. Wenn es nicht länger Verweise auf ein Objekt gibt, wird es beim nächsten Ausführen der automatischen Speicherbereinigung entfernt und der entsprechende Speicherplatz wird freigegeben. Mit dem folgenden Code werden beispielsweise ein neues Objekt erstellt und dem Objekt ein Verweis auf die `myObject`-Variable zugewiesen:

```
var myObject:Object = new Object();
```

Solange ein Verweis auf das Objekt vorhanden ist, wird der Speicherplatz, den das Objekt belegt, bei der automatischen Speicherbereinigung nicht freigegeben. Wenn der Wert von `myObject` so geändert wird, dass es auf ein anderes Objekt verweist oder auf den Wert `null` gesetzt ist, wird der vom ursprünglichen Objekt belegte Speicherplatz bei der Speicherbereinigung freigegeben. Dies erfolgt jedoch nur, wenn keine anderen Verweise auf das ursprüngliche Objekt vorliegen.

Wenn Sie `myObject` als Schlüssel in einem Dictionary-Objekt verwenden, erstellen Sie damit einen weiteren Verweis auf das ursprüngliche Objekt. Mit dem folgenden Code werden beispielsweise zwei Verweise auf ein Objekt erstellt, die Variable `myObject` und der Schlüssel im `myMap`-Objekt:

```
import flash.utils.Dictionary;

var myObject:Object = new Object();
var myMap:Dictionary = new Dictionary();
myMap[myObject] = "foo";
```

Wenn das Objekt, auf das `myObject` verweist, bei der Speicherbereinigung entfernt werden soll, müssen Sie alle Verweise auf dieses Objekt löschen. In diesem Fall müssen Sie den Wert von `myObject` ändern und den Schlüssel `myObject` in `myMap` löschen, wie im folgenden Codebeispiel dargestellt:

```
myObject = null;
delete myMap[myObject];
```

Wahlweise können Sie mit dem `useWeakReference`-Parameter des Dictionary-Konstruktors alle Wörterbuchschlüssel in *schwache Verweise* ändern. Bei der Speicherbereinigung werden schwache Verweise ignoriert, d. h. ein Objekt mit ausschließlich schwachen Verweisen wird bei der Speicherbereinigung entfernt. Im folgenden Codebeispiel müssen Sie den Schlüssel `myObject` in `myMap` nicht entfernen, damit das Objekt bei der Speicherbereinigung entfernt wird:

```
import flash.utils.Dictionary;

var myObject:Object = new Object();
var myMap:Dictionary = new Dictionary(true);
myMap[myObject] = "foo";
myObject = null; // Make object eligible for garbage collection.
```

Mehrdimensionale Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mehrdimensionale Arrays enthalten als Elemente andere Arrays, Nehmen wir als Beispiel eine Aufgabenliste, die als indiziertes String-Array gespeichert ist:

```
var tasks:Array = ["wash dishes", "take out trash"];
```

Wenn Sie für jeden Wochentag eine separate Aufgabenliste speichern möchten, können Sie ein mehrdimensionales Array mit jeweils einem Element für jeden Wochentag erstellen. Jedes Element enthält ein indiziertes Array, ähnlich dem `tasks`-Array, in dem die Aufgabenliste gespeichert ist. Sie können in mehrdimensionalen Arrays jede Kombination aus indizierten und assoziativen Arrays verwenden. Bei den Beispielen in den folgenden Abschnitten werden entweder zwei indizierte Arrays oder ein assoziatives Array mit indizierten Arrays verwendet. Sie können zu Übungszwecken andere Kombinationen verwenden.

Zwei indizierte Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie zwei indizierte Arrays verwenden, können Sie das Ergebnis in einer Tabelle anzeigen. Die Elemente des ersten Arrays entsprechen den Zeilen der Tabelle und die Elemente des zweiten Arrays den Spalten.

Im folgenden mehrdimensionalen Array wird beispielsweise mit zwei indizierten Arrays eine Aufgabenliste für jeden Wochentag geführt. Das erste Array, `masterTaskList`, wird mit dem Konstruktor der Array-Klasse erstellt. Jedes Element des Arrays entspricht einem Wochentag, dabei steht die Indexposition 0 für Montag und die Indexposition 6 für Sonntag. Diese Elemente können Sie sich als Zeilen einer Tabelle vorstellen. Sie können eine Aufgabenliste für jeden Tag erstellen, indem Sie jedem der sieben Elemente ein Array-Literal zuweisen, das Sie im `masterTaskList`-Array erstellen. Die Array-Literale entsprechen den Spalten in der Tabelle.

```
var masterTaskList:Array = new Array();
masterTaskList[0] = ["wash dishes", "take out trash"];
masterTaskList[1] = ["wash dishes", "pay bills"];
masterTaskList[2] = ["wash dishes", "dentist", "wash dog"];
masterTaskList[3] = ["wash dishes"];
masterTaskList[4] = ["wash dishes", "clean house"];
masterTaskList[5] = ["wash dishes", "wash car", "pay rent"];
masterTaskList[6] = ["mow lawn", "fix chair"];
```

Mit dem Array-Zugriffsoperator (`[]`) können Sie auf die einzelnen Einträge in den Aufgabenlisten zugreifen. Das erste Klammerpaar gibt den Wochentag an, das zweite Klammerpaar die Aufgabenliste für den entsprechenden Tag. Wenn Sie beispielsweise die zweite Aufgabe der Liste für Mittwoch abrufen möchten, verwenden Sie die Indexposition 2 für Mittwoch und dann die Indexposition 1 für die zweite Aufgabe in der Liste.

```
trace(masterTaskList[2][1]); // output: dentist
```

Wenn Sie die erste Aufgabe der Liste für Sonntag abrufen möchten, verwenden Sie die Indexposition 6 für Sonntag und die Indexposition 0 für die erste Aufgabe in der Liste.

```
trace(masterTaskList[6][0]); // output: mow lawn
```

Assoziatives Array mit einem indizierten Array

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Damit leichter auf die einzelnen Arrays zugegriffen werden kann, können Sie ein assoziatives Array für die Wochentage und ein indiziertes Array für die Aufgabenlisten verwenden. Durch Verwendung eines assoziativen Arrays muss nicht mit der Punktsyntax auf einen bestimmten Wochentag verwiesen werden, dies jedoch auf Kosten zusätzlicher Verarbeitungszeit beim Zugreifen auf alle Elemente des assoziativen Arrays. Im folgenden Beispiel wird ein assoziatives Array als Grundlage für eine Aufgabenliste verwendet, mit einem Schlüssel-Wert-Paar für jeden Wochentag:

```
var masterTaskList:Object = new Object();
masterTaskList["Monday"] = ["wash dishes", "take out trash"];
masterTaskList["Tuesday"] = ["wash dishes", "pay bills"];
masterTaskList["Wednesday"] = ["wash dishes", "dentist", "wash dog"];
masterTaskList["Thursday"] = ["wash dishes"];
masterTaskList["Friday"] = ["wash dishes", "clean house"];
masterTaskList["Saturday"] = ["wash dishes", "wash car", "pay rent"];
masterTaskList["Sunday"] = ["mow lawn", "fix chair"];
```

Durch die Punktsyntax wird der Code leserlicher, da so verschachtelte Klammerpaare vermieden werden können.

```
trace(masterTaskList.Wednesday[1]); // output: dentist
trace(masterTaskList.Sunday[0]);// output: mow lawn
```

Sie können die Aufgabenliste mit einer `for..in`-Schleife durchlaufen, müssen jedoch anstelle der Punktsyntax den Array-Zugriffsoperator (`[]`) verwenden, um auf den jedem Schlüssel zugeordneten Wert zuzugreifen. Da `masterTaskList` ein assoziatives Array ist, werden die Elemente nicht unbedingt in der erwarteten Reihenfolge abgerufen, wie im folgenden Beispiel dargestellt:

```
for (var day:String in masterTaskList)
{
    trace(day + ": " + masterTaskList[day])
}
/* output:
Sunday: mow lawn,fix chair
Wednesday: wash dishes,dentist,wash dog
Friday: wash dishes,clean house
Thursday: wash dishes
Monday: wash dishes,take out trash
Saturday: wash dishes,wash car,pay rent
Tuesday: wash dishes,pay bills
*/
```

Klonen von Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Array-Klasse verfügt über keine integrierte Methode zum Erstellen von Kopien von Arrays. Sie können eine *oberflächlicheKopie* eines Arrays erstellen, indem Sie entweder die `concat()`-Methode oder die `slice()`-Methode ohne Argumente aufrufen. Wenn das ursprüngliche Array über Objekte als Elemente verfügt, werden bei einer oberflächlichen Kopie nur die Verweise auf die Objekte und nicht die Objekte selbst kopiert. Die Kopie verweist auf die gleichen Objekte wie das ursprüngliche Array. Alle an den Objekten vorgenommenen Änderungen spiegeln sich in beiden Arrays wider.

Bei einer *tiefen Kopie* werden alle im ursprünglichen Array gefundenen Objekte ebenfalls kopiert, sodass das neue Array nicht auf die gleichen Objekte wie das ursprüngliche Array verweist. Bei einer tiefen Kopie müssen mehrere Codezeilen angegeben werden. Dies spricht in der Regel für die Erstellung einer Funktion. Diese Funktion kann als globale Dienstprogrammfunktion oder als Methode einer Array-Unterklasse erstellt werden.

Im folgenden Beispiel wird die `clone()`-Funktion definiert, mit der tiefe Kopien erstellt werden. Der Algorithmus stammt aus einer häufig verwendeten Java-Programmiertechnik. Mit dieser Funktion wird eine tiefe Kopie erstellt, indem das Array in eine Instanz der ByteArray-Klasse serialisiert und das Array dann in ein neues Array geschrieben wird. Dieser Funktion können Objekte übergeben werden, sodass sie sowohl bei indizierten als auch bei assoziativen Arrays eingesetzt werden kann, wie im folgenden Codebeispiel dargestellt:

```
import flash.utils.ByteArray;

function clone(source:Object):*
{
    var myBA:ByteArray = new ByteArray();
    myBA.writeObject(source);
    myBA.position = 0;
    return(myBA.readObject());
}
```

Erweitern der Array-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Array-Klasse ist eine der wenigen Hauptklassen, die nicht endgültig festgelegt („final“) ist. Dies bedeutet, dass Sie Unterklassen der Array-Klasse erstellen können. In diesem Abschnitt wird an einem Beispiel erläutert, wie eine Unterklasse der Array-Klasse erstellt wird. Darüber hinaus werden einige Probleme erörtert, die unter Umständen auftreten können.

Wie weiter oben erwähnt, sind Arrays in ActionScript nicht typisiert, Sie können jedoch eine Array-Unterklasse erstellen, die nur Elemente eines bestimmten Datentyps enthalten kann. In dem Beispiel in den folgenden Abschnitten wird eine Array-Unterklasse mit dem Namen „TypedArray“ erstellt, bei der die Elemente auf Werte des im ersten Parameter angegebenen Datentyps beschränkt sind. Die TypedArray-Klasse wird hier lediglich exemplarisch zur Erweiterung der Array-Klasse verwendet und eignet sich möglicherweise aus mehreren Gründen nicht für die Praxis. Erstens erfolgt die Typüberprüfung zur Laufzeit und nicht bei der Kompilierung. Zweitens werden Diskrepanzen ignoriert, die in der TypedArray-Methode auftreten können, und keine Ausnahmen ausgelöst. Die Methoden können jedoch auf einfache Weise so geändert werden, dass Ausnahmen ausgelöst werden. Mit der Klasse kann drittens nicht verhindert werden, dass über den Array-Zugriffsoperator Werte jedes Datentyps in das Array eingefügt werden. Darüber hinaus wurde beim Programmierstil auf Einfachheit statt auf Leistungsoptimierung gesetzt.

Hinweis: *Sie können ein Typ-Array mit den hier beschriebenen Techniken erstellen. Es empfiehlt sich jedoch, stattdessen ein Vector-Objekt zu verwenden. Eine Vector-Instanz ist ein Typ-Array in Reinform und bietet im Vergleich mit der Array-Klasse oder den Unterklassen Leistungssteigerungen und weitere Vorteile. In diesem Abschnitt soll die Erstellung einer Array-Unterklasse veranschaulicht werden.*

Deklarieren der Unterklasse

Geben Sie mit dem Schlüsselwort `extends` an, dass es sich um eine Unterklasse der Array-Klasse handelt. Bei einer Array-Unterklasse sollte wie bei der Array-Klasse das `dynamic`-Attribut verwendet werden. Andernfalls kann die Unterklasse nicht ordnungsgemäß verwendet werden.

Im folgenden Code ist die Definition der TypedArray-Klasse dargestellt. Sie enthält eine Konstante für den Datentyp, eine Konstruktormethode und die vier Methoden zum Hinzufügen von Elementen zum Array. Der Code für die einzelnen Methoden ist im folgenden Codebeispiel nicht angegeben, er wird jedoch in den folgenden Abschnitten getrennt und vollständig erläutert.

```
public dynamic class TypedArray extends Array
{
    private const dataType:Class;

    public function TypedArray(...args) {}

    AS3 override function concat(...args):Array {}

    AS3 override function push(...args):uint {}

    AS3 override function splice(...args) {}

    AS3 override function unshift(...args):uint {}
}
```

Bei allen vier überschriebenen Methoden wird der AS3-Namespace anstelle des `public`-Attributs verwendet, da in diesem Beispiel davon ausgegangen wird, dass die Compileroption `-as3` auf `true` und die Compileroption `-es` auf `false` gesetzt ist. Dies sind die Standardeinstellungen für Adobe Flash Builder und AdobeFlashProfessional.

Erfahrene Entwickler, die die Verwendung der Prototypvererbung bevorzugen, können zwei kleine Änderungen an der TypedArray-Klasse vornehmen, sodass diese mit der auf `true` gesetzten Compileroption `-es` kompiliert werden kann. Entfernen Sie zunächst alle Vorkommen des `override`-Attributs und ersetzen Sie den AS3-Namespace durch das `public`-Attribut. Ersetzen Sie dann alle vier Vorkommen von `super` durch `Array.prototype`.

TypedArray-Konstruktor

Der Konstruktor der Unterklasse stellt eine interessante Herausforderung dar, da er eine Liste mit Argumenten beliebiger Länge annehmen muss. Die Herausforderung besteht in der Weise, auf die die Argumente an den Superkonstruktor übergeben werden, um das Array zu erstellen. Wenn Sie die Liste der Argumente als Array übergeben, wird sie im Superkonstruktor als einzelnes Argument des Array-Typs erkannt, und das resultierende Array hat immer die Länge eines Elements. Normalerweise werden übergebene Argumentlisten mithilfe der `Function.apply()`-Methode verarbeitet, bei der ein Array mit Argumenten als zweiter Parameter verwendet wird, das Array jedoch beim Ausführen der Funktion in eine Liste mit Argumenten konvertiert wird. Leider kann die `Function.apply()`-Methode bei Konstruktoren nicht verwendet werden.

Die einzige verbleibende Möglichkeit besteht darin, den Code des Array-Konstruktors im TypedArray-Konstruktor neu zu erstellen. Im folgenden Codebeispiel ist der im Konstruktor der Array-Klasse verwendete Algorithmus dargestellt, den Sie im Konstruktor der Array-Unterklasse wiederverwenden können:

```
public dynamic class Array
{
    public function Array(...args)
    {
        var n:uint = args.length
        if (n == 1 && (args[0] is Number))
        {
            var dlen:Number = args[0];
            var ulen:uint = dlen;
            if (ulen != dlen)
            {
                throw new RangeError("Array index is not a 32-bit unsigned integer ("+dlen+")");
            }
            length = ulen;
        }
        else
        {
            length = n;
            for (var i:int=0; i < n; i++)
            {
                this[i] = args[i]
            }
        }
    }
}
```

Für den TypedArray-Konstruktor wird im Wesentlichen der gleiche Code verwendet wie beim Array-Konstruktor, es werden nur vier Änderungen am Code vorgenommen. Die Parameterliste enthält erstens einen neuen erforderlichen Parameter des Class-Typs, mit dem der Datentyp des Arrays angegeben werden kann. Der an den Konstruktor übergebene Datentyp wird zweitens der `dataType`-Variablen zugewiesen. In der `else`-Anweisung wird drittens der Wert der `length`-Eigenschaft nach der `for`-Schleife zugewiesen, sodass `length` nur Argumente mit dem korrekten Datentyp enthält. Im Rumpf der `for`-Schleife wird viertens die überschriebene Version der `push()`-Methode verwendet, sodass ausschließlich Argumente mit dem korrekten Datentyp zum Array hinzugefügt werden. Im folgenden Beispiel ist die TypedArray-Konstruktorfunktion dargestellt:

```
public dynamic class TypedArray extends Array
{
    private var dataType:Class;
    public function TypedArray(typeParam:Class, ...args)
    {
        dataType = typeParam;
        var n:uint = args.length
        if (n == 1 && (args[0] is Number))
        {
            var dlen:Number = args[0];
            var ulen:uint = dlen
            if (ulen != dlen)
            {
                throw new RangeError("Array index is not a 32-bit unsigned integer ("+dlen+")")
            }
            length = ulen;
        }
        else
        {
            for (var i:int=0; i < n; i++)
            {
                // type check done in push()
                this.push(args[i])
            }
            length = this.length;
        }
    }
}
```

Überschriebene Methoden der TypedArray-Klasse

Bei der TypedArray-Klasse werden die vier Methoden der Array-Klasse überschrieben, mit denen Elemente zu einem Array hinzugefügt werden können. Bei allen überschriebenen Methoden wird jeweils eine Typüberprüfung eingefügt, über die verhindert wird, dass Elemente mit einem ungültigen Datentyp hinzugefügt werden. Anschließend wird bei allen Methoden die Superclass-Version der jeweiligen Methode aufgerufen.

Die `push()`-Methode durchläuft die Liste der Argumente mit einer `for..in`-Schleife und führt für jedes Argument eine Typüberprüfung durch. Alle Argumente, die nicht den korrekten Datentyp aufweisen, werden mit der `splice()`-Methode aus dem `args`-Array entfernt. Nach Abschluss der `for..in`-Schleife enthält das `args`-Array nur Werte mit dem Typ `dataType`. Die Superclass-Version von `push()` wird dann mit dem aktualisierten `args`-Array aufgerufen, wie im folgenden Codebeispiel dargestellt:

```
AS3 override function push(...args):uint
{
    for (var i:* in args)
    {
        if (!(args[i] is dataType))
        {
            args.splice(i,1);
        }
    }
    return (super.push.apply(this, args));
}
```

Mit der `concat()`-Methode wird das temporäre TypedArray-Array mit dem Namen `passArgs` erstellt, in dem die Argumente nach erfolgreicher Typüberprüfung gespeichert werden. Dadurch kann der in der `push()`-Methode vorhandene Code für die Typüberprüfung erneut verwendet werden. Mit einer `for..in`-Schleife wird das `args`-Array durchlaufen. Anschließend wird `push()` für jedes Argument aufgerufen. Da `passArgs` als TypedArray typisiert ist, wird die TypedArray-Version von `push()` ausgeführt. Mit der `concat()`-Methode wird dann die zugehörige Superclass-Version aufgerufen, wie im folgenden Codebeispiel dargestellt:

```
AS3 override function concat(...args):Array
{
    var passArgs:TypedArray = new TypedArray(dataType);
    for (var i:* in args)
    {
        // type check done in push()
        passArgs.push(args[i]);
    }
    return (super.concat.apply(this, passArgs));
}
```

Bei der `splice()`-Methode wird eine beliebig lange Liste mit Argumenten verwendet. Die ersten beiden Argumente verweisen jedoch immer auf eine Indexposition und die Anzahl der zu löschenden Elemente. Aus diesem Grund wird mit der überschriebenen `splice()`-Methode nur eine Typüberprüfung der Elemente des `args`-Arrays ab Indexposition 2 durchgeführt. Im Codebeispiel wird in der `for`-Schleife scheinbar ein rekursiver Aufruf von `splice()` ausgeführt. Es handelt sich jedoch um keinen rekursiven Aufruf, da `args` den Array-Typ und nicht den TypedArray-Typ aufweist. Dies bedeutet, dass beim Aufrufen von `args.splice()` die Superclass-Version der Methode aufgerufen wird. Nach Abschluss der `for..in`-Schleife enthält das `args`-Array ab Indexposition 2 nur Werte mit dem korrekten Datentyp. Mit `splice()` wird zudem die zugehörige Superclass-Version aufgerufen, wie im folgenden Codebeispiel dargestellt:

```
AS3 override function splice(...args):*
{
    if (args.length > 2)
    {
        for (var i:int=2; i< args.length; i++)
        {
            if (!(args[i] is dataType))
            {
                args.splice(i,1);
            }
        }
    }
    return (super.splice.apply(this, args));
}
```

Bei der `unshift()`-Methode, mit der Elemente am Anfang eines Arrays hinzugefügt werden, kann ebenfalls eine beliebige Liste mit Argumenten übergeben werden. Bei der überschriebenen `unshift()`-Methode wird ein ähnlicher Algorithmus wie bei der `push()`-Methode verwendet, wie im folgenden Codebeispiel dargestellt:

```
    AS3 override function unshift(...args):uint
    {
        for (var i:* in args)
        {
            if (!(args[i] is dataType))
            {
                args.splice(i,1);
            }
        }
        return (super.unshift.apply(this, args));
    }
}
```

Arrays-Beispiel: PlayList

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Im Beispiel „PlayList“ werden Verfahren zum Verwenden von Arrays im Kontext einer Anwendung verdeutlicht, mit der eine Wiedergabeliste mit Musiktiteln verwaltet wird. Dies sind im Einzelnen:

- Erstellen eines indizierten Arrays
- Hinzufügen von Elementen zu einem indizierten Array
- Sortieren eines Arrays von Objekten nach verschiedenen Eigenschaften mit verschiedenen Sortieroptionen
- Konvertieren eines Arrays in einen durch Zeichen getrennten String

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „PlayList“ befinden sich im Ordner „Samples/PlayList“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
PlayList.mxml oder PlayList.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/playlist/PlayList.as	Eine Klasse, die eine Liste mit Musiktiteln darstellt. Die Klasse verwendet ein Array zum Speichern der Liste und verwaltet die Sortierung der Listeneinträge.
com/example/programmingas3/playlist/Song.as	Ein Wertobjekt mit Informationen zu einem einzelnen Musiktitel. Die Elemente, die mit der PlayList-Klasse verwaltet werden, sind Song-Instanzen.
com/example/programmingas3/playlist/SortProperty.as	Eine Pseudoaufzählung, deren verfügbare Werte die Eigenschaften der Song-Klasse darstellen, nach denen eine Liste mit Song-Objekten sortiert werden kann.

Überblick über die PlayList-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der PlayList-Klasse wird eine Gruppe von Song-Objekten verwaltet. Sie verfügt über öffentliche Methoden mit Funktionen zum Hinzufügen eines Musiktitels zur Wiedergabeliste (die `addSong()`-Methode) und zum Sortieren der Musiktitel in der Liste (die `sortList()`-Methode). Darüber hinaus enthält die Klasse die schreibgeschützte Accessor-Eigenschaft `songList`, mit der auf die eigentlichen Musiktitel in der Wiedergabeliste zugegriffen werden kann. Intern werden die Musiktitel der PlayList-Klasse mit einer privaten Array-Variablen protokolliert:

```
public class PlayList
{
    private var _songs:Array;
    private var _currentSort:SortProperty = null;
    private var _needToSort:Boolean = false;
    ...
}
```

Neben der Array-Variablen `_songs` zum Protokollieren der Liste mit Musiktiteln der PlayList-Klasse wird mit zwei weiteren privaten Variablen protokolliert, ob die Liste sortiert werden muss (`_needToSort`) und nach welcher Eigenschaft die Liste zu einem bestimmten Zeitpunkt sortiert wird (`_currentSort`).

Wie bei allen Objekten ist mit der Deklaration einer Array-Instanz nur ein Teil der Erstellung eines Arrays erledigt. Vor dem Zugreifen auf die Eigenschaften oder Methoden einer Array-Instanz muss diese instanziiert werden. Dies erfolgt über den Konstruktor der PlayList-Klasse.

```
    public function PlayList()
    {
        this._songs = new Array();
        // Set the initial sorting.
        this.sortList(SortProperty.TITLE);
    }
```

Mit der ersten Zeile des Konstruktors wird die `_songs`-Variable instanziiert, sodass sie verwendet werden kann. Darüber hinaus wird die `sortList()`-Methode aufgerufen, um die ursprüngliche Sortiereigenschaft festzulegen.

Hinzufügen eines Musiktitels zur Liste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn ein Benutzer einen neuen Musiktitel in die Anwendung eingibt, wird mit dem Code im Dateneingabeformular die `addSong()`-Methode der PlayList-Klasse aufgerufen.

```
    /**
     * Adds a song to the playlist.
     */
    public function addSong(song:Song):void
    {
        this._songs.push(song);
        this._needToSort = true;
    }
```

In `addSong()` wird die `push()`-Methode des `_songs`-Arrays aufgerufen und das an `addSong()` übergebene Song-Objekt wird als neues Element in dieses Array eingefügt. Mit der `push()`-Methode wird das neue Element am Ende des Arrays eingefügt, unabhängig von zuvor durchgeführten Sortiervorgängen. Dies bedeutet, dass die Liste der Musiktitel nach dem Aufruf der `push()`-Methode wahrscheinlich nicht mehr korrekt sortiert ist, sodass die `_needToSort`-Variable auf `true` gesetzt wird. Theoretisch kann die `sortList()`-Methode sofort aufgerufen werden, sodass nicht protokolliert werden muss, ob die Liste zu einem bestimmten Zeitpunkt sortiert oder nicht sortiert ist. Praktisch muss die Liste mit Musiktiteln jedoch erst vor dem Abrufen sortiert werden. Durch Zurückstellen des Sortiervorgangs wird keine unnötige Sortierung durchgeführt, wenn beispielsweise mehrere Musiktitel zur Liste hinzugefügt werden, bevor die Liste abgerufen wird.

Sortieren der Liste mit Musiktiteln

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Da es sich bei den über die Wiedergabeliste verwalteten Song-Instanzen um komplexe Objekte handelt, möchten Benutzer der Anwendung die Wiedergabeliste möglicherweise nach verschiedenen Eigenschaften sortieren, beispielsweise nach Musiktitel oder Veröffentlichungsjahr. In der Anwendung „PlayList“ setzt sich die Sortierung der Liste mit Musiktiteln aus drei Teilen zusammen: Identifizieren der Eigenschaft, nach der die Liste sortiert werden soll, Angeben der erforderlichen Sortieroptionen beim Sortieren nach dieser Eigenschaft und Durchführen des eigentlichen Sortiervorgangs.

Eigenschaften für die Sortierung

Ein Song-Objekt protokolliert mehrere Eigenschaften, einschließlich Musiktitel, Interpret, Veröffentlichungsjahr, Dateiname und benutzerdefinierter Genres, denen der Musiktitel zugeordnet wird. Nur die ersten drei Eigenschaften erweisen sich als praktisch für die Sortierung. Als Annehmlichkeit für Entwickler umfasst das Beispiel die SortProperty-Klasse, die als Aufzählung der Werte dient, mit denen die verfügbaren Sortiereigenschaften angegeben werden.

```
public static const TITLE:SortProperty = new SortProperty("title");
public static const ARTIST:SortProperty = new SortProperty("artist");
public static const YEAR:SortProperty = new SortProperty("year");
```

Die SortProperty-Klasse enthält die drei Konstanten `TITLE`, `ARTIST` und `YEAR`, in denen jeweils ein String mit dem Namen der zugeordneten Eigenschaft der Song-Klasse gespeichert ist, die für die Sortierung verwendet werden kann. Im gesamten restlichen Code wird bei jeder Angabe einer Sortiereigenschaft das Aufzählungselement verwendet. Im PlayList-Konstruktor wird die Liste beispielsweise anfänglich durch Aufrufen der `sortList()`-Methode sortiert:

```
// Set the initial sorting.
this.sortList(SortProperty.TITLE);
```

Da die Eigenschaft zum Sortieren durch `SortProperty.TITLE` angegeben wird, werden die Musiktitel nach Titel sortiert.

Sortieren nach Eigenschaft und Festlegen von Sortieroptionen

Die Sortierung der Liste mit Musiktiteln erfolgt über die `sortList()`-Methode der PlayList-Klasse:

```
/**
 * Sorts the list of songs according to the specified property.
 */
public function sortList(sortProperty:SortProperty):void
{
    ...
    var sortOptions:uint;
    switch (sortProperty)
    {
        case SortProperty.TITLE:
            sortOptions = Array.CASEINSENSITIVE;
            break;
        case SortProperty.ARTIST:
            sortOptions = Array.CASEINSENSITIVE;
            break;
        case SortProperty.YEAR:
            sortOptions = Array.NUMERIC;
            break;
    }

    // Perform the actual sorting of the data.
    this._songs.sortOn(sortProperty.propertyName, sortOptions);

    // Save the current sort property.
    this._currentSort = sortProperty;

    // Record that the list is sorted.
    this._needToSort = false;
}
```

Bei der Sortierung nach Titel oder Interpret empfiehlt sich die alphabetische Sortierung. Bei der Sortierung nach Jahr empfiehlt sich jedoch die numerische Sortierung. Mit der `switch`-Anweisung wird die geeignete Sortieroption, die in der Variablen `sortOptions` gespeichert ist, entsprechend dem im `sortProperty`-Parameter angegebenen Wert definiert. Hier wird erneut anhand der benannten Aufzählungselemente und nicht nach hartkodierten Werten zwischen den einzelnen Eigenschaften unterschieden.

Wenn die Sortiereigenschaft und die Sortieroptionen festgelegt sind, wird das `_songs`-Array durch Aufrufen der zugehörigen `sortOn()`-Methode sortiert. Dabei werden diese beiden Werte als Parameter übergeben. Die aktuelle Sortiereigenschaft sowie die Tatsache, dass die Wiedergabeliste derzeit sortiert ist, werden protokolliert.

Kombinieren von Array-Elementen in einem durch Zeichen getrennten String

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Neben der Verwendung eines Arrays zum Verwalten der Wiedergabeliste in der PlayList-Klasse werden in diesem Beispiel auch Arrays in der Song-Klasse verwendet, um die Liste der Genres zu verwalten, denen ein bestimmter Musiktitel zugeordnet ist. Betrachten Sie das folgende Codesegment zur Definition der Song-Klasse:

```
private var _genres:String;

public function Song(title:String, artist:String, year:uint, filename:String, genres:Array)
{
    ...
    // Genres are passed in as an array
    // but stored as a semicolon-separated string.
    this._genres = genres.join(";");
}
```

Beim Erstellen einer neuen Song-Instanz wird der `genres`-Parameter, mit dem das Genre (oder die Genres) des Musiktitels angegeben wird, als Array-Instanz definiert. Dadurch können mehrere Genres auf einfache Weise in einer einzelnen Variablen gruppiert werden, die dann an den Konstruktor übergeben werden kann. Intern werden die Genres jedoch in der privaten `_genres`-Variablen der Song-Klasse als durch Semikola getrennte String-Instanzen verwaltet. Der Array-Parameter wird in einen durch Semikola getrennten String konvertiert, indem die zugehörige `join()`-Methode mit dem Literalstring `„;"` als festgelegtem Trennzeichen aufgerufen wird.

Ebenso können mit den `genres`-Zugriffsmethoden Genres als Array festgelegt oder abgerufen werden:

```
public function get genres():Array
{
    // Genres are stored as a semicolon-separated String,
    // so they need to be transformed into an Array to pass them back out.
    return this._genres.split(";");
}
public function set genres(value:Array):void
{
    // Genres are passed in as an array,
    // but stored as a semicolon-separated string.
    this._genres = value.join(";");
}
```

Die `set`-Zugriffsmethode von `genres` entspricht in ihrer Funktionsweise dem Konstruktor: Ihr wird ein Array übergeben, und sie ruft die `join()`-Methode auf, um das Array in einen durch Semikola getrennten String zu konvertieren. Die `get`-Zugriffsmethode führt den umgekehrten Vorgang aus: Die `split()`-Methode der `_genres`-Variablen wird aufgerufen. Sie unterteilt den String in ein Array von Werten mit dem angegebenen Trennzeichen (wie zuvor der Literalstring `„;"`).

Kapitel 4: Verarbeiten von Fehlern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die „Verarbeitung“ eines Fehlers bedeutet, die Anwendung mit Logik auszustatten, die auf den Fehler reagieren oder den Fehler beheben kann. Fehler treten entweder bei der Kompilierung einer Anwendung oder beim Ausführen einer kompilierten Anwendung auf. Beim Verarbeiten von Fehlern durch eine Anwendung geschieht *etwas* als Reaktion auf einen aufgetretenen Fehler. Im Gegensatz dazu erfolgt keine Reaktion, wenn der den Fehler verursachende Prozess ohne Meldung beendet wird. Wenn die Fehlerverarbeitung korrekt eingesetzt wird, erleichtert sie das Schützen der Anwendung und ihrer Benutzer vor andernfalls unerwartetem Verhalten.

Fehlerverarbeitung ist jedoch ein sehr breites Gebiet, das viele Fehlerarten abdecken muss, die während des Kompilierens oder beim Ausführen einer Anwendung auftreten. Der Schwerpunkt dieses Abschnitts liegt auf Laufzeitfehlern (die beim Ausführen einer Anwendung ausgegeben werden), den verschiedenen Fehlertypen, die generiert werden können, und den Vorteilen des Fehlerverarbeitungssystems in ActionScript 3.0.

Grundlagen der Fehlerverarbeitung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei einem Laufzeitfehler tritt im ActionScript-Code ein Problem auf, in dessen Folge der ActionScript-Inhalt nicht mehr ausgeführt werden kann. Um die problemlose Ausführung des ActionScript-Codes für Benutzer sicherzustellen, fügen Sie in die Anwendung Code ein, der solche Fehler verarbeitet (korrigiert, vermeidet oder zumindest dem Benutzer die genaue Ursache meldet). Dieser Vorgang wird als *Fehlerverarbeitung* bezeichnet.

Fehlerverarbeitung ist ein sehr breites Gebiet, das viele Fehlerarten abdecken muss, die während des Kompilierens oder beim Ausführen einer Anwendung auftreten. Fehler, die zur Kompilierzeit auftreten, sind oft einfacher zu finden. Korrigieren Sie diese Fehler, damit die SWF-Datei erstellt werden kann.

Laufzeitfehler sind schwieriger zu erkennen, da der fehlerhafte Code ausgeführt werden muss, damit sie auftreten. Wenn ein Bereich des Programms mehrere Codezweige enthält, z. B. in einer `if..then..else`-Anweisung, testen Sie jede mögliche Bedingung mit allen real möglichen Eingabewerten, um zu bestätigen, dass der Code fehlerfrei ist.

Laufzeitfehler lassen sich in zwei Kategorien einteilen: *Programmfehler*: Fehler im ActionScript-Code, z. B. durch Angeben eines falschen Datentyps für einen Methodenparameter; *logische Fehler*: Fehler in der Logik (Datenüberprüfung und Werteverarbeitung) des Programms, z. B. durch Verwenden einer fehlerhaften Formel zum Berechnen von Zinswerten in einer Bankanwendung. Beide Fehlertypen können häufig durch konsequentes Testen der Anwendung rechtzeitig erkannt und korrigiert werden.

Das Idealziel besteht darin, alle Fehler in der Anwendung zu erkennen und zu entfernen, bevor diese für Endbenutzer veröffentlicht wird. Es können jedoch nicht alle Fehler vorhergesehen oder vermieden werden. Angenommen, Ihre ActionScript-Anwendung lädt Informationen von einer bestimmten Website, die Sie nicht testen können. Wenn diese Website einmal nicht verfügbar ist, funktioniert der Teil Ihrer Anwendung nicht ordnungsgemäß, der von den externen Daten abhängt. Der wichtigste Aspekt der Fehlerverarbeitung ist, die Anwendung auf solche unbekannten Situationen vorzubereiten, damit sie angemessen darauf reagieren kann. Die Benutzer müssen in der Lage sein, weiterhin mit der Anwendung zu arbeiten. Zumindest sollten sie jedoch in einer Fehlermeldung informiert werden, warum die Anwendung nicht mehr funktioniert.

Laufzeitfehler werden in ActionScript auf zwei Arten repräsentiert:

- Error-Klassen: Vielen Fehlern ist eine entsprechende Fehlerklasse zugeordnet. Beim Auftreten eines Fehlers erzeugt die Flash-Laufzeitumgebung (wie Flash Player oder Adobe AIR) eine Instanz der spezifischen Fehlerklasse, die diesem bestimmten Fehler zugeordnet ist. Im Programmcode können die in diesem Fehlerobjekt enthaltenen Informationen ausgewertet werden, um angemessen auf den Fehler reagieren zu können.
- Fehlerereignisse: Manchmal tritt ein Fehler auf, wenn die Flash-Laufzeitumgebung normalerweise ein Ereignis auslösen würden. In diesen Fällen wird stattdessen ein Fehlerereignis ausgelöst. Jedem Fehlerereignis ist eine Klasse zugeordnet. Die Flash-Laufzeitumgebung übergibt eine Instanz dieser Klasse an die Methoden, die für das Fehlerereignis registriert sind.

Um festzustellen, ob eine bestimmte Methode einen Fehler oder ein Fehlerereignis auslösen kann, lesen Sie den Eintrag zu der betreffenden Methode im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Wichtige Konzepte und Begriffe

Die folgende Referenzliste enthält wichtige Begriffe für die Programmierung von Routinen für die Fehlerverarbeitung:

Asynchron Ein Programmbefehl wie beispielsweise ein Methodenaufruf, der kein sofortiges Ergebnis zurückgibt. Stattdessen wird das Ergebnis (bzw. ein Fehler) als Ereignis zurückgegeben.

Abfangen Wenn eine Ausnahme (ein Laufzeitfehler) auftritt und der Code diese Ausnahme erkennt, wird dies als *Abfangen* der Ausnahme im Code bezeichnet. Nachdem eine Ausnahme abgefangen wurde, wird anderer ActionScript-Code in der Flash-Laufzeitumgebung nicht mehr über die Ausnahme benachrichtigt

Debugger-Version Eine spezielle Version der Flash-Laufzeit, zum Beispiel die Debugger-Version von Flash Player oder der AIR Debug Launcher (ADL), die Code zum Benachrichtigen von Benutzern über Laufzeitfehler enthält. In der Standardversion von Flash Player oder Adobe AIR (die von den meisten Benutzern verwendet wird) werden die nicht mit dem ActionScript-Code verarbeiteten Fehler ignoriert. In den Debugger-Versionen (die in Adobe Flash CS4 Professional und Adobe Flash Builder enthalten sind) wird beim Auftreten eines nicht verarbeiteten Fehlers eine Warnmeldung angezeigt.

Ausnahmebedingung Ein Fehler, der beim Ausführen einer Anwendung auftritt und der in der Flash-Laufzeit nicht automatisch behoben werden kann.

Erneutes Auslösen Wenn der Code eine Ausnahme abfängt, benachrichtigt die Flash-Laufzeit andere Objekte nicht mehr über die Ausnahme. Wenn es wichtig ist, dass auch andere Objekte die Ausnahme erhalten, muss die Ausnahme im Code *erneut ausgelöst* werden, damit der Benachrichtigungsvorgang neu gestartet wird.

Synchron Ein Programmbefehl wie beispielsweise ein Methodenaufruf, der ein sofortiges Ergebnis zurückgibt (oder sofort einen Fehler auslöst). Das Ergebnis kann im selben Codeblock verwendet werden.

Auslösen Das Benachrichtigen einer Flash-Laufzeit (und damit auch von anderen Objekten und ActionScript-Code) über das Auftreten eines Fehlers wird als *Auslösen* eines Fehlers bezeichnet.

Fehlertypen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Entwickeln und Ausführen von Anwendungen treffen Sie auf unterschiedliche Fehlertypen und Fehlerbezeichnungen. In der folgenden Liste werden die wichtigsten Fehlertypen und Begriffe vorgestellt.

- *Kompilierzeitfehler* werden vom ActionScript-Compiler während der Codekompilierung ausgelöst. Diese Fehler treten auf, wenn Syntaxprobleme im Programmcode das Erstellen der Anwendung verhindern.

- *Laufzeitfehler* treten beim Ausführen einer Anwendung auf, nachdem diese kompiliert wurde. Es handelt sich dabei um Fehler, die auftreten, während eine SWF-Datei in einer Flash-Laufzeitumgebung (wie Adobe Flash Player oder Adobe AIR) wiedergegeben wird. In den meisten Fällen verarbeiten Sie Laufzeitfehler bei ihrem Auftreten, indem Sie sie an den Benutzer melden und entsprechende Schritte unternehmen, damit die Anwendung dennoch weiter ausgeführt werden kann. Wenn es sich um einen schwerwiegenden Fehler handelt, etwa wenn keine Verbindungen mit Websites hergestellt oder erforderliche Daten nicht geladen werden können, ist es mithilfe der Fehlerverarbeitung möglich, die Anwendung kontrolliert zu beenden.
- *Synchrone Fehler* sind Laufzeitfehler, die gleichzeitig mit dem Aufrufen einer Funktion auftreten, beispielsweise wenn Sie eine bestimmte Methode verwenden und das übergebene Argument ungültig ist, sodass die Flash-Laufzeitumgebung eine Ausnahme auslöst. Die meisten Fehler sind synchron, d. h. sie treten zum Zeitpunkt der Ausführung der Anweisung auf, und die Steuerung geht sofort auf die geeignetste `catch`-Anweisung über.

  Beispielsweise wird mit dem folgenden Codeauszug ein Laufzeitfehler ausgelöst, da die `browse()`-Methode nicht aufgerufen wird, bevor das Programm versucht, eine Datei hochzuladen:

```
var fileRef:FileReference = new FileReference();
try
{
    fileRef.upload(new URLRequest("http://www.yourdomain.com/fileupload.cfm"));
}
catch (error:IllegalOperationError)
{
    trace(error);
    // Error #2037: Functions called in incorrect sequence, or earlier
    // call was unsuccessful.
}
```

  In diesem Fall wird ein synchroner Laufzeitfehler ausgelöst, da in Flash Player festgestellt wurde, dass die `browse()`-Methode nicht aufgerufen wurde, bevor der Dateiuploadversuch erfolgt ist.

  Detaillierte Informationen zur synchronen Fehlerverarbeitung finden Sie unter „Verarbeiten synchroner Fehler in einer Anwendung“ auf Seite 62.

- *AsynchroneFehler* sind Laufzeitfehler, die außerhalb des normalen Programmflusses auftreten. Sie generieren Ereignisse, die von Ereignis-Listenern abgefangen werden. Bei einem asynchronen Vorgang initiiert eine Funktion einen Vorgang, wartet jedoch nicht auf dessen Abschluss. Sie können Fehlerereignis-Listener erstellen, die überwachen, ob die Anwendung oder der Benutzer einen bestimmten Vorgang ausführt. Wenn dieser Vorgang fehlschlägt, fangen Sie den Fehler mit dem Ereignis-Listener ab und reagieren auf das Fehlerereignis. Der Ereignis-Listener ruft dann eine Ereignisprozedurfunktion auf, um auf geeignete Weise auf den Fehler zu reagieren. Mit der Ereignisprozedur kann beispielsweise ein Dialogfeld angezeigt werden, in dem der Benutzer zum Beheben des Fehlers aufgefordert wird.

  Rufen Sie sich das zuvor vorgestellte Beispiel für einen synchronen Fehler beim Dateiupload in Erinnerung. Wenn Sie vor dem Start des Dateiuploads die `browse()`-Methode aufrufen, löst Flash Player verschiedene Ereignisse aus. Beispielsweise wird beim Start des Uploads das `open`-Ereignis ausgelöst. Nach dem erfolgreichen Beenden des Dateiuploads wird das `complete`-Ereignis ausgelöst. Da die Ereignisverarbeitung asynchron ist (d. h. nicht zu bestimmten, bekannten Zeitpunkten erfolgt), verwenden Sie zum Warten auf diese Ereignisse die `addEventListener()`-Methode, wie im folgenden Code dargestellt:

```
var fileRef:FileReference = new FileReference();
fileRef.addEventListener(Event.SELECT, selectHandler);
fileRef.addEventListener(Event.OPEN, openHandler);
fileRef.addEventListener(Event.COMPLETE, completeHandler);
fileRef.browse();

function selectHandler(event:Event):void
{
    trace("...select...");
    var request:URLRequest = new URLRequest("http://www.yourdomain.com/fileupload.cfm");
    request.method = URLRequestMethod.POST;
    event.target.upload(request);
}
function openHandler(event:Event):void
{
    trace("...open...");
}
function completeHandler(event:Event):void
{
    trace("...complete...");
}
```

Detaillierte Informationen zur asynchronen Fehlerverarbeitung finden Sie unter „Reagieren auf Fehlerereignisse und Status“ auf Seite 67.

- *Nicht abgefangene Ausnahmen* sind Fehler, die ausgelöst werden und für die keine entsprechende Programmlogik zur angemessenen Reaktion (beispielsweise eine `catch`-Anweisung) vorliegt. Wenn in einer Anwendung ein Fehler ausgelöst wird und auf der aktuellen oder einer höheren Ebene keine geeignete `catch`-Anweisung oder keine Ereignisprozedur zur Fehlerverarbeitung gefunden wird, gilt dieser Fehler als nicht abgefangene Ausnahme.

  Wenn ein nicht abgefangener Fehler auftritt, löst die Laufzeit ein `uncaughtError`-Ereignis aus. Dieses Ereignis wird auch „globale Fehlerprozedur“ genannt. Dieses Ereignis wird vom UncaughtErrorEvents-Objekt der SWF ausgelöst, das über die `LoaderInfo.uncaughtErrorEvents`-Eigenschaft zur Verfügung steht. Wenn keine Listener für das `uncaughtError`-Ereignis registriert sind, ignoriert die Laufzeit nicht abgefangene Fehler. Die Laufzeit versucht, die Ausführung fortzusetzen, sofern die SWF durch den Fehler nicht gestoppt wird.

  Zusätzlich zum Auslösen des `uncaughtError`-Ereignisses reagieren Debugger-Versionen der Flash-Laufzeit auf nicht abgefangene Fehler, indem das aktuelle Skript beendet wird. Dann wird der nicht abgefangene Fehler in der Ausgabe einer `trace`-Anweisung angezeigt oder eine Fehlermeldung wird in eine Protokolldatei geschrieben. Wenn das Ausnahmeobjekt eine Instanz der Error-Klasse oder einer ihrer Unterklassen ist, werden in der Ausgabe auch Stacktrace-Informationen angezeigt. Weitere Informationen zur Verwendung der Debugger-Version von Flash-Laufzeitumgebungen finden Sie unter „Arbeiten mit den Debugger-Versionen der Flash-Laufzeitumgebungen“ auf Seite 61.

  Hinweis: *Wenn bei der Verarbeitung eines uncaughtError-Ereignisses ein Fehlerereignis von einer uncaughtError-Prozedur ausgelöst wird, wird die Ereignisprozedur mehrmals aufgerufen. Dies führt zu einer unendlichen Ausnahmeschleife. Ein solches Szenario sollte vermieden werden.*

Fehlerverarbeitung in ActionScript 3.0

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Da viele Anwendungen ausgeführt werden können, ohne dass die Programmlogik zum Verarbeiten von Fehlern erstellt wurde, neigen Entwickler dazu, die Programmierung der Fehlerverarbeitung für ihre Anwendungen hinauszuzögern. Ohne Fehlerverarbeitung kann es jedoch leicht vorkommen, dass Benutzer durch Anwendungen eingeschränkt oder frustriert werden, wenn Vorgänge nicht wie erwartet durchgeführt werden können. ActionScript 2.0 verfügt über eine Error-Klasse, mit der Sie entsprechende Programmlogik als benutzerdefinierte Funktionen erstellen können, die Ausnahmen mit einer speziellen Meldung auslösen. Da die Fehlerverarbeitung zum Erstellen benutzerfreundlicher Anwendungen unabdingbar ist, enthält ActionScript 3.0 eine erweiterte Architektur zum Abfangen von Fehlern.

Hinweis: *Im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform werden zwar die Ausnahmen dokumentiert, die von zahlreichen Methoden ausgelöst werden, ein Anspruch auf Vollständigkeit wird jedoch nicht erhoben. Möglicherweise löst eine Methode eine Ausnahme wegen Syntaxfehlern oder anderen Problemen aus, die nicht explizit in der Methodenbeschreibung genannt sind, selbst wenn andere Ausnahmen für diese Methode aufgeführt sind.*

Elemente der Fehlerverarbeitung in ActionScript 3.0

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 enthält viele Tools zur Fehlerverarbeitung. Dazu gehören u.a.:

- Error-Klassen. ActionScript 3.0 umfasst eine breite Palette an Error-Klassen, um den Umfang der Situationen auszuweiten, in denen Fehlerobjekte auftreten können. Jede der Error-Klassen erleichtert es, in Anwendungen bestimmte Fehlerzustände zu verarbeiten und auf sie zu reagieren. Diese können sich auf Systemfehler (z. B. MemoryError-Fehler), Programmierfehler (z. B. ArgumentError-Fehler), Netzwerk- und Kommunikationsfehler (z. B. URIError-Fehler) oder andere Fehlersituationen beziehen. Weitere Informationen zu den einzelnen Klassen finden Sie unter „Vergleich der Error-Klassen“ auf Seite 71.
- Weniger stillschweigendes Fehlschlagen. In früheren Versionen von Flash Player wurden Fehler nur erzeugt und gemeldet, wenn explizit die `throw`-Anweisung verwendet wurde. Für Flash Player 9 und höhere Flash-Laufzeitumgebungen lösen native ActionScript-Methoden und -Eigenschaften Laufzeitfehler aus. Diese Fehler ermöglichen es Ihnen, die aufgetretenen Ausnahmen effizient zu verarbeiten und individuell auf einzelne Ausnahmen zu reagieren.
- Aussagekräftige Fehlermeldungen während des Debuggens. Beim Einsatz der Debugger-Version einer Flash-Laufzeitumgebung werden für problematischen Code oder Fehlersituationen umfangreiche Fehlermeldungen erzeugt, mit denen Sie schnell den Grund für das Fehlschlagen eines bestimmten Codeblocks ermitteln können. Dank dieser Meldungen können Fehler effizienter behoben werden. Weitere Informationen finden Sie unter „Arbeiten mit den Debugger-Versionen der Flash-Laufzeitumgebungen“ auf Seite 61.
- Genau zuordenbare Fehler ermöglichen eindeutige Fehlermeldungen, die Benutzern angezeigt werden. In früheren Versionen von Flash Player gab die `FileReference.upload()`-Methode den booleschen Wert `false` zurück, wenn der Aufruf von `upload()` nicht erfolgreich war. Dieser eine Wert diente zum Kennzeichnen von fünf verschiedenen möglichen Fehlern. Wenn beim Aufrufen der `upload()`-Methode in ActionScript 3.0 ein Fehler auftritt, bieten Ihnen vier spezifische Fehler die Möglichkeit, genauere Fehlermeldungen für Endbenutzer anzuzeigen.

- Gezieltere Fehlerverarbeitung. Für viele häufig auftretende Fehlersituationen werden eindeutige Fehler ausgelöst. Beispielsweise hat in ActionScript 2.0 vor dem Füllen eines FileReference-Objekts mit Daten die `name`-Eigenschaft den Wert `null`. Deshalb müssen Sie vor dem Verwenden oder Anzeigen der `name`-Eigenschaft sicherstellen, dass der Wert gesetzt und nicht `null` ist. In ActionScript 3.0 wird bei dem Versuch, vor dem Füllen des Objekts mit Daten auf die `name`-Eigenschaft zuzugreifen, in Flash Player oder AIR ein IllegalOperationError-Fehler ausgelöst, der Sie darüber informiert, dass der Wert noch nicht gesetzt wurde. Sie können `try..catch..finally`-Blöcke verwenden, um den Fehler zu verarbeiten. Weitere Informationen finden Sie unter „Verwenden von „try..catch..finally“-Anweisungen“ auf Seite 62.
- Keine spürbaren Leistungseinbußen. Verwenden von `try..catch..finally`-Blöcken zur Fehlerverarbeitung nimmt im Vergleich zu früheren ActionScript-Versionen nur geringe oder gar keine zusätzlichen Ressourcen in Anspruch.
- Eine ErrorEvent-Klasse, mit deren Hilfe Sie Listener für bestimmte asynchrone Fehlerereignisse erstellen können. Weitere Informationen finden Sie unter „Reagieren auf Fehlerereignisse und Status“ auf Seite 67.

Fehlerverarbeitungsstrategien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Solange in einer Anwendung keine Probleme auftreten, kann sie meist erfolgreich ausgeführt werden, selbst wenn Sie keine Programmlogik zur Fehlerverarbeitung in den Code integrieren. Wenn Sie Fehler jedoch nicht aktiv verarbeiten und in der Anwendung ein Problem auftritt, bleibt es den Benutzern verborgen, warum die Anwendung in diesem Fall plötzlich fehlschlägt.

Es gibt unterschiedliche Möglichkeiten für die Fehlerverarbeitung in Anwendungen. In der nachstehenden Liste sind die drei wichtigsten Optionen für die Fehlerverarbeitung zusammengefasst:

- Verwenden von `try..catch..finally`-Anweisungen. Mit diesen Anweisungen werden synchrone Fehler zum Zeitpunkt ihres Auftretens abgefangen. Sie können diese Anweisungen hierarchisch verschachteln, um Ausnahmen auf unterschiedlichen Ebenen der Codeausführung abzufangen. Weitere Informationen finden Sie unter „Verwenden von „try..catch..finally“-Anweisungen“ auf Seite 62.
- Erstellen benutzerdefinierter Fehlerobjekte. Sie können die Error-Klasse verwenden, um benutzerdefinierte Fehlerobjekte zu definieren und mit ihnen bestimmte Vorgänge in der Anwendung zu verfolgen, die von den integrierten Fehlertypen nicht abgedeckt werden. Dann können Sie `try..catch..finally`-Anweisungen für Ihre benutzerdefinierten Fehlerobjekte verwenden. Weitere Informationen finden Sie unter „Erstellen benutzerdefinierter Fehlerklassen“ auf Seite 66.
- Programmieren von Ereignis-Listenern und Ereignisprozeduren zur Reaktion auf Fehlerereignisse. Durch diese Strategie können Sie globale Fehlerprozeduren erstellen, mit deren Hilfe Sie ähnliche Ereignisse verarbeiten können, ohne in `try..catch..finally`-Blöcken unnötig viel Code wiederholen zu müssen. Bei diesem Ansatz ist außerdem die Wahrscheinlichkeit größer, asynchrone Fehler abfangen zu können. Weitere Informationen finden Sie unter „Reagieren auf Fehlerereignisse und Status“ auf Seite 67.

Arbeiten mit den Debugger-Versionen der Flash-Laufzeitumgebungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Adobe stellt für Entwickler spezielle Versionen der Flash-Laufzeitumgebungen bereit, um das Debugging von Anwendungen zu unterstützen. Wenn Sie Adobe Flash Professional oder Adobe Flash Builder installieren, erhalten Sie auch die Debugger-Version von Flash Player. Sie erhalten auch ein Dienstprogramm für das Debugging von Adobe AIR-Anwendungen, wenn Sie eines dieser Tools installieren, oder als Teil des Adobe AIR SDK. Dieses Dienstprogramm wird ADL genannt.

Die Debugger-Versionen und die normalen Versionen von Flash Player und Adobe AIR unterscheiden sich deutlich bei der Anzeige von Fehlern. In den Debugger-Versionen wird der Fehlertyp (z. B. Error, IOError oder EOFError), die Fehlernummer und eine Fehlermeldung im Klartext angezeigt. In den normalen Versionen wird nur der Fehlertyp und die Fehlernummer angezeigt. Betrachten Sie den folgenden Beispielcode:

```
try
{
    tf.text = myByteArray.readBoolean();
}
catch (error:EOFError)
{
    tf.text = error.toString();
}
```

Wenn die `readBoolean()`-Methode in der Debugger-Version von Flash Player einen EOFError-Fehler auslöst, wird die folgende Meldung im `tf`-Textfeld angezeigt: „EOFError: Fehler #2030: Unerwartetes Dateiende aufgetreten.“

In der normalen Version von Flash Player oder Adobe AIR wird dagegen beim gleichen Fehler folgender Text angezeigt: „EOFError: Error #2030“.

Hinweis: *Die Debugger-Player senden ein Ereignis namens „allComplete“; vermeiden Sie die Erstellung von benutzerdefinierten Ereignissen mit dem Namen „allComplete“. Andernfalls kann beim Debugging ein unerwartetes Verhalten auftreten.*

Um in den normalen Versionen die Ressourcennutzung und die Dateigröße zu minimieren, werden keine Fehlermeldungsstrings angezeigt. Sie können die Fehlernummer in der Dokumentation nachschlagen, um die entsprechende Fehlermeldung zu finden (in den Anhängen im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform). Alternativ können Sie auch den Fehler mit den Debugger-Versionen von Flash Player und AIR reproduzieren, damit die vollständige Meldung angezeigt wird.

Verarbeiten synchroner Fehler in einer Anwendung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die häufigste Fehlerverarbeitung besteht aus Programmlogik für synchrone Fehler. Dabei fügen Sie im Code Anweisungen ein, um synchrone Fehler beim Ausführen einer Anwendung abzufangen. Mit dieser Art der Fehlerverarbeitung kann die Anwendung beim Fehlschlagen von Funktionen Laufzeitfehler feststellen und die Abarbeitung fortsetzen. Die Logik zum Abfangen synchroner Fehler umfasst `try..catch..finally`-Anweisungen, mit denen Vorgänge probeweise ausgeführt, eventuelle Fehlerreaktionen der Flash-Laufzeitumgebung abgefangen und schließlich bei Bedarf andere Vorgänge durchgeführt werden, um einen fehlgeschlagenen Vorgang zu verarbeiten.

Verwenden von „try..catch..finally“-Anweisungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie mit synchronen Laufzeitfehlern arbeiten, verwenden Sie die `try..catch..finally`-Anweisungen zum Abfangen von Fehlern. Bei einem Laufzeitfehler gibt die Flash-Laufzeitumgebung einen Ausnahmefehler aus. Dies bedeutet, dass Flash Player die normale Ausführung unterbricht und ein spezielles Objekt mit dem Error-Typ erstellt. Das Error-Objekt wird dann an den ersten verfügbaren `catch`-Block übergeben.

Die `try`-Anweisung umschließt Anweisungen, die möglicherweise Fehler verursachen können. Die `catch`-Anweisung wird stets mit einer `try`-Anweisung verwendet. Wenn in einer Anweisung im `try`-Anweisungsblock ein Fehler auftritt, werden die `catch`-Anweisungen ausgeführt, die dieser `try`-Anweisung zugewiesen sind.

Die `finally`-Anweisung umschließt Anweisungen, die unabhängig davon ausgeführt werden, ob im `try`-Block ein Fehler aufgetreten ist. Wenn kein Fehler auftritt, werden nach Abschluss der Anweisungen im `try`-Block die Anweisungen innerhalb des `finally`-Blocks ausgeführt. Beim Auftreten eines Fehlers werden zuerst die entsprechende `catch`-Anweisung und dann die Anweisungen im `finally`-Block ausgeführt.

Der nachstehende Code veranschaulicht die Syntax für die Verwendung von `try..catch..finally`-Anweisungen:

```
try
{
    // some code that could throw an error
}
catch (err:Error)
{
    // code to react to the error
}
finally
{
    // Code that runs whether an error was thrown. This code can clean
    // up after the error, or take steps to keep the application running.
}
```

Jede `catch`-Anweisung kann einen bestimmten Ausnahmetyp verarbeiten. In `catch`-Anweisungen können nur Fehlerklassen angegeben werden, die Unterklassen der Error-Klasse sind. Jede `catch`-Anweisung wird in der angegebenen Reihenfolge überprüft. Nur die erste `catch`-Anweisung, die mit dem Typ des ausgelösten Fehlers übereinstimmt, wird ausgeführt. Wenn Sie erst die übergeordnete Error-Klasse und dann eine Unterklasse der Error-Klasse testen, wird daher nur eine Übereinstimmung mit der übergeordneten Error-Klasse erkannt. Dies wird mit dem folgenden Code veranschaulicht:

```
try
{
    throw new ArgumentError("I am an ArgumentError");
}
catch (error:Error)
{
    trace("<Error> " + error.message);
}
catch (error:ArgumentError)
{
    trace("<ArgumentError> " + error.message);
}
```

Der vorstehende Code führt zu folgender Ausgabe:

```
<Error> I am an ArgumentError
```

Um „ArgumentError“ korrekt abzufangen, stellen Sie sicher, dass die spezifischeren Fehlertypen zuerst und die allgemeineren Fehlertypen später aufgeführt sind, wie im folgenden Code gezeigt:

```
try
{
    throw new ArgumentError("I am an ArgumentError");
}
catch (error:ArgumentError)
{
    trace("<ArgumentError> " + error.message);
}
catch (error:Error)
{
    trace("<Error> " + error.message);
}
```

Verschiedene Methoden und Eigenschaften der ActionScript-API lösen Laufzeitfehler aus, wenn bei der Ausführung Fehler auftreten. Beispielsweise löst die `close()`-Methode der Sound-Klasse einen IOError aus, wenn der Audio-Stream mit der Methode nicht geschlossen werden kann, wie im folgenden Code dargestellt ist:

```
var mySound:Sound = new Sound();
try
{
    mySound.close();
}
catch (error:IOError)
{
    // Error #2029: This URLStream object does not have an open stream.
}
```

Wenn Sie mit dem ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform vertraut werden, merken Sie, welche Methoden Ausnahmen auslösen, wie in den Beschreibungen der einzelnen Methoden erläutert.

throw-Anweisung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Flash-Laufzeitumgebungen lösen Ausnahmen aus, wenn in Ihrer ausgeführten Anwendung Probleme auftreten. Zusätzlich können Sie selbst mithilfe der `throw`-Anweisung explizit Ausnahmen auslösen. Beim expliziten Auslösen von Fehlern wird empfohlen, Instanzen der Error-Klasse oder einer ihrer Unterklassen auszulösen. Im folgenden Code wird eine `throw`-Anweisung veranschaulicht, die eine Instanz der Error-Klasse (`MyErr`) auslöst und schließlich eine Funktion (`myFunction()`) aufruft, um auf den ausgelösten Fehler zu reagieren:

```
var MyError:Error = new Error("Encountered an error with the numUsers value", 99);
var numUsers:uint = 0;
try
{
    if (numUsers == 0)
    {
        trace("numUsers equals 0");
    }
}
catch (error:uint)
{
    throw MyError; // Catch unsigned integer errors.
}
catch (error:int)
{
    throw MyError; // Catch integer errors.
}
catch (error:Number)
{
    throw MyError; // Catch number errors.
}
catch (error:*)
{
    throw MyError; // Catch any other error.
}
finally
{
    myFunction(); // Perform any necessary cleanup here.
}
```

Beachten Sie, dass die `catch`-Anweisungen so angeordnet sind, dass die spezifischsten Datentypen zuerst aufgeführt sind. Wenn die `catch`-Anweisung für den Number-Datentyp an erster Stelle steht, werden die catch-Anweisungen für die Datentypen „uint“ und „int“ nie ausgeführt.

Hinweis: *In der Programmiersprache Java muss jede Funktion, die eine Ausnahme auslösen kann, dies zuvor deklarieren. Dazu müssen die auslösbaren Ausnahmeklassen in einer `throws`-Klausel aufgeführt sein, die der Funktionsdeklaration angefügt ist. In ActionScript ist es nicht erforderlich, die Ausnahmen zu deklarieren, die von einer Funktion ausgelöst werden.*

Anzeigen einer einfachen Fehlermeldung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Einer der größten Vorteile des neuen Ausnahme- und Fehlerereignismodells ist, dass Sie den Benutzern mitteilen können, dass und warum ein Vorgang fehlgeschlagen ist. Ihre Aufgabe besteht darin, den Code zum Anzeigen der Meldung zu programmieren und mögliche Reaktionen anzubieten.

Der folgende Code zeigt eine einfache `try..catch`-Anweisung zum Anzeigen des Fehlers in einem Textfeld:

```
package
{
    import flash.display.Sprite;
    import flash.text.TextField;

    public class SimpleError extends Sprite
    {
        public var employee:XML =
            <EmpCode>
                <costCenter>1234</costCenter>
                <costCenter>1-234</costCenter>
            </EmpCode>;

        public function SimpleError()
        {
            try
            {
                if (employee.costCenter.length() != 1)
                {
                    throw new Error("Error, employee must have exactly one cost center assigned.");
                }
            }
            catch (error:Error)
            {
                var errorMessage:TextField = new TextField();
                errorMessage.autoSize = TextFieldAutoSize.LEFT;
                errorMessage.textColor = 0xFF0000;
                errorMessage.text = error.message;
                addChild(errorMessage);
            }
        }
    }
}
```

Mit einem breiteren Spektrum von Fehlerklassen und integrierten Compilerfehlern bietet ActionScript 3.0 umfassendere Informationen als die Vorgängerversionen von ActionScript über die Gründe für das Fehlschlagen von Vorgängen. Diese Informationen ermöglichen es Ihnen, stabilere Anwendungen mit besserer Fehlerverarbeitung zu erstellen.

Erneutes Auslösen von Fehlern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Erstellen von Anwendungen gibt es mehrere Situationen, in denen Sie einen Fehler erneut auslösen müssen, wenn dieser nicht ordnungsgemäß verarbeitet werden kann. Der folgende Code zeigt z. B. einen verschachtelten `try..catch`-Block, der einen benutzerdefinierten ApplicationError-Fehler auslöst, wenn der verschachtelte `catch`-Block den Fehler nicht verarbeiten kann:

```
try
{
    try
    {
        trace("<< try >>");
        throw new ApplicationError("some error which will be rethrown");
    }
    catch (error:ApplicationError)
    {
        trace("<< catch >> " + error);
        trace("<< throw >>");
        throw error;
    }
    catch (error:Error)
    {
        trace("<< Error >> " + error);
    }
}
catch (error:ApplicationError)
{
    trace("<< catch >> " + error);
}
```

Mit diesem Codebeispiel wird folgende Ausgabe generiert:

```
<< try >>
<< catch >> ApplicationError: some error which will be rethrown
<< throw >>
<< catch >> ApplicationError: some error which will be rethrown
```

Der verschachtelte `try`-Block löst einen benutzerdefinierten ApplicationError-Fehler aus, der mit dem nachfolgenden `catch`-Block abgefangen wird. In diesem verschachtelten `catch`-Block wird versucht, den Fehler zu verarbeiten. Wenn dies nicht gelingt, wird das ApplicationError-Objekt an den übergeordneten `try..catch`-Block übergeben.

Erstellen benutzerdefinierter Fehlerklassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können eine der Error-Standardklassen erweitern, um in ActionScript eigene spezielle Klassen zu erstellen. Es gibt eine Reihe von Gründen, eigene Fehlerklassen zu erstellen:

- Identifizieren spezifischer Fehler oder Fehlergruppen, die nur in Ihrer Anwendung vorkommen.

  Beispielsweise können Sie für die von Ihrem Code ausgelösten Fehler – zusätzlich zu den von einer Flash-Laufzeitumgebung abgefangenen Fehlern – abweichende Aktionen ausführen. Sie können eine Unterklasse der Error-Klasse erstellen, um den neuen Fehlertyp in `try..catch`-Blöcken zu verfolgen.

- Bereitstellen eigener Fehleranzeigefunktionen für die in Ihrer Anwendung erzeugten Fehler.

  Sie können beispielsweise eine neue `toString()`-Methode erstellen, mit der Fehlermeldungen auf bestimmte Weise formatiert werden. Sie können zudem eine `lookupErrorString()`-Methode definieren, der ein Fehlercode übergeben wird und die entsprechend der Sprachauswahl des Benutzers die richtige Fehlermeldung abruft.

Spezialisierte Fehlerklassen müssen die ActionScript-Kernklasse Error erweitern. Es folgt ein Beispiel für eine spezialisierte AppError-Klasse, die die Error-Klasse erweitert:

```
public class AppError extends Error
{
    public function AppError(message:String, errorID:int)
    {
        super(message, errorID);
    }
}
```

Im Folgenden finden Sie ein Beispiel zur Verwendung von AppError in eigenen Projekten:

```
try
{
    throw new AppError("Encountered Custom AppError", 29);
}
catch (error:AppError)
{
    trace(error.errorID + ": " + error.message)
}
```

Hinweis: *Wenn Sie die `Error.toString()`-Methode in Ihrer Unterklasse überschreiben möchten, versehen Sie sie mit einem `...(rest)`-Parameter. In der ECMAScript-Sprachspezifikation, auf der ActionScript 3.0 basiert, wird die `Error.toString()`-Methode auf diese Weise definiert, und ActionScript 3.0 definiert sie aus Gründen der Abwärtskompatibilität genauso. Verwenden Sie daher beim Überschreiben der `Error.toString()`-Methode exakt dieselben Parameter. Es ist nicht nötig, Parameter zur Laufzeit an die `toString()`-Methode zu übergeben, da diese ignoriert werden.*

Reagieren auf Fehlerereignisse und Status

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine der spürbarsten Verbesserungen bei der Fehlerverarbeitung in ActionScript 3.0 ist die Unterstützung für die Fehlerereignisverarbeitung, um auf asynchrone Fehler beim Ausführen einer Anwendung reagieren zu können. (Definitionen für asynchrone Fehler finden Sie unter „Fehlertypen“ auf Seite 56.)

Sie können Ereignis-Listener und Ereignisprozeduren erstellen, um auf Fehlerereignisse zu reagieren. Viele Klassen lösen Fehlerereignisse auf dieselbe Weise wie andere Ereignisse aus. Beispielsweise lösen Instanzen der XMLSocket-Klasse normalerweise drei Ereignistypen aus: `Event.CLOSE`, `Event.CONNECT` und `DataEvent.DATA`. Beim Auftreten eines Problems kann die XMLSocket-Klasse jedoch auch die Ereignisse `IOErrorEvent.IOError` und `SecurityErrorEvent.SECURITY_ERROR` auslösen. Weitere Informationen zu Ereignis-Listenern und Ereignisprozeduren finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Fehlerereignisse gehören einer der beiden folgenden Kategorien an:

- Fehlerereignisse, die die ErrorEvent-Klasse erweitern

  Die flash.events.ErrorEvent-Klasse enthält die Eigenschaften und Methoden zum Verwalten von Fehlern im Zusammenhang mit Netzwerk- oder Kommunikationsvorgängen, die in einer laufenden Anwendung auftreten. Die AsyncErrorEvent-, IOErrorEvent- und SecurityErrorEvent-Klassen erweitern die ErrorEvent-Klasse. Wenn Sie die Debugger-Version einer Flash-Laufzeitumgebung verwenden, werden Sie zur Laufzeit in Dialogfeldern über alle aufgetretenen Fehlerereignisse informiert, für die keine Listener-Funktion vorhanden ist.

- Statusbasierte Fehlerereignisse

  Die statusbasierten Fehlerereignisse beziehen sich auf die Eigenschaften `netStatus` und `status` der Netzwerk- und Kommunikationsklassen. Wenn eine Flash-Laufzeitumgebung beim Lesen oder Schreiben von Daten ein Problem auftritt, wird der Wert der `netStatus.info.level`-Eigenschaft oder der `status.level`-Eigenschaft (je nach verwendetem Klassenobjekt) auf `error` gesetzt. Sie können diesen Fehler in der Ereignisprozedurfunktion abfragen, indem Sie überprüfen, ob die `level`-Eigenschaft den Wert `error` hat.

Verwenden von Fehlerereignissen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die ErrorEvent-Klasse und ihre Unterklassen enthalten Fehlertypen zum Verarbeiten von Fehlern, die in einer Flash-Laufzeitumgebung bei dem Versuch ausgelöst werden, Daten zu lesen oder zu schreiben.

Im folgenden Beispiel werden sowohl eine `try..catch`-Anweisung als auch Fehlerereignisprozeduren verwendet, um Fehler anzuzeigen, die beim Lesen einer lokalen Datei möglicherweise auftreten. An den mit „eigenen Fehlerverarbeitungscode hier einfügen“ gekennzeichneten Stellen können Sie anspruchsvolleren Verarbeitungscode einfügen, um Benutzern Optionen bereitzustellen oder aber Fehler automatisch zu verarbeiten:

```
package
{
    import flash.display.Sprite;
    import flash.errors.IOError;
    import flash.events.IOErrorEvent;
    import flash.events.TextEvent;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.net.URLRequest;
    import flash.text.TextField;
    import flash.text.TextFieldAutoSize;

    public class LinkEventExample extends Sprite
    {
        private var myMP3:Sound;
        public function LinkEventExample()
        {
            myMP3 = new Sound();
            var list:TextField = new TextField();
            list.autoSize = TextFieldAutoSize.LEFT;
            list.multiline = true;
            list.htmlText = "<a href=\"event:track1.mp3\">Track 1</a><br>";
            list.htmlText += "<a href=\"event:track2.mp3\">Track 2</a><br>";
            addEventListener(TextEvent.LINK, linkHandler);
            addChild(list);
        }
```

```
        private function playMP3(mp3:String):void
        {
            try
            {
                myMP3.load(new URLRequest(mp3));
                myMP3.play();
            }
            catch (err:Error)
            {
                trace(err.message);
                // your error-handling code here
            }
            myMP3.addEventListener(IOErrorEvent.IO_ERROR, errorHandler);
        }

        private function linkHandler(linkEvent:TextEvent):void
        {
            playMP3(linkEvent.text);
            // your error-handling code here
        }

        private function errorHandler(errorEvent:IOErrorEvent):void
        {
            trace(errorEvent.text);
            // your error-handling code here
        }
    }
}
```

Verwenden von Statusänderungsereignissen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash-Laufzeitumgebungen ändern den Wert der `netStatus.info.level`- oder `status.level`-Eigenschaften für die Klassen, die die `level`-Eigenschaft unterstützen, während der Ausführung der Anwendung automatisch. Folgende Klassen verfügen über die `netStatus.info.level`-Eigenschaft: NetConnection, NetStream und SharedObject. Die Klassen HTTPStatusEvent, Camera, Microphone und LocalConnection weisen die `status.level`-Eigenschaft auf. Sie können eine Prozedurfunktion programmieren, um auf Änderungen des Werts von `level` zu reagieren und Kommunikationsfehler zu verfolgen.

Im folgenden Beispiel wird mit einer `netStatusHandler()`-Funktion der Wert der `level`-Eigenschaft getestet. Wenn die `level`-Eigenschaft angibt, dass ein Fehler aufgetreten ist, wird die Meldung „Video stream failed“ (Video-Stream fehlgeschlagen) ausgegeben.

```
package
{
    import flash.display.Sprite;
    import flash.events.NetStatusEvent;
    import flash.events.SecurityErrorEvent;
    import flash.media.Video;
    import flash.net.NetConnection;
    import flash.net.NetStream;

    public class VideoExample extends Sprite
    {
        private var videoUrl:String = "Video.flv";
        private var connection:NetConnection;
        private var stream:NetStream;

        public function VideoExample()
        {
            connection = new NetConnection();
            connection.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
            connection.addEventListener(SecurityErrorEvent.SECURITY_ERROR, securityErrorHandler);
            connection.connect(null);
        }

        private function netStatusHandler(event:NetStatusEvent):void
        {
            if (event.info.level == "error")
            {
                trace("Video stream failed")
            }
            else
            {
                connectStream();
            }
        }

        private function securityErrorHandler(event:SecurityErrorEvent):void
        {
            trace("securityErrorHandler: " + event);
        }

        private function connectStream():void
        {
            var stream:NetStream = new NetStream(connection);
            var video:Video = new Video();
            video.attachNetStream(stream);
            stream.play(videoUrl);
            addChild(video);
        }
    }
}
```

Vergleich der Error-Klassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript stellt eine Reihe vordefinierter Error-Klassen bereit. Sie können jedoch auch dieselben Error-Klassen in Ihrem eigenen Code verwenden. Es gibt zwei Haupttypen von Fehlerklassen in ActionScript 3.0: Error-Kernklassen und Error-Klassen des flash.error-Pakets. Das flash.error-Paket enthält zusätzliche Klassen, die Entwicklung und Debugging von ActionScript 3.0-Anwendungen erleichtern sollen.

Error-Kernklassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zu den Error-Kernklassen gehören die Klassen Error, ArgumentError, EvalError, RangeError, ReferenceError, SecurityError, SyntaxError, TypeError, URIError und VerifyError. Jede dieser Klassen befindet sich im Namespace der obersten Ebene.

Klassenname	Beschreibung	Hinweise
Error	Mit der Error-Klasse können Ausnahmen ausgelöst werden. Sie ist die Basisklasse für die anderen in ECMAScript definierten Ausnahmeklassen: EvalError, RangeError, ReferenceError, SyntaxError, TypeError und URIError.	Die Error-Klasse dient als Basisklasse für alle Laufzeitfehler. Sie ist auch die empfohlene Basisklasse für alle benutzerdefinierten Fehlerklassen.
ArgumentError	Die ArgumentError-Klasse repräsentiert einen Fehler, der auftritt, wenn die in einem Funktionsaufruf übergebenen Parameterwerte nicht den für diese Funktion definierten Parametern entsprechen.	Beispiele für solche Fehler: • Einer Methode werden zu wenige oder zu viele Argumente übergeben. • Es wurde erwartet, dass ein Argument Mitglied einer Aufzählung ist. Dies war jedoch nicht der Fall.
EvalError	EvalError-Ausnahmen werden ausgelöst, wenn Parameter an den Konstruktor der Function-Klasse übergeben werden oder wenn die `eval()`-Funktion im Benutzercode aufgerufen wird.	In ActionScript 3.0 wird die `eval()`-Funktion nicht mehr unterstützt. Versuche, diese Funktion zu verwenden, führen zu einem Fehler. In älteren Versionen von Flash Player wurde die `eval()`-Funktion eingesetzt, um auf Variablen, Eigenschaften, Objekte oder Movieclips über deren Namen zuzugreifen.
RangeError	Eine RangeError-Ausnahme wird ausgelöst, wenn ein numerischer Wert außerhalb eines zulässigen Bereichs liegt.	Beispielsweise wird eine RangeError-Ausnahme durch die Timer-Klasse ausgelöst, wenn ein Verzögerungswert negativ oder keine endliche Zahl ist. RangeError-Ausnahmen werden auch ausgelöst, wenn Sie versuchen, Anzeigeobjekte in einer ungültigen Tiefe hinzuzufügen.
ReferenceError	Eine ReferenceError-Ausnahme wird ausgelöst, wenn bei einem versiegelten (nicht dynamischen) Objekt versucht wird, auf eine nicht definierte Eigenschaft zu verweisen. In den ActionScript-Compilerversionen vor ActionScript 3.0 wurde kein Fehler bei dem Versuch ausgelöst, auf eine Eigenschaft mit dem Wert `undefined` zuzugreifen. In ActionScript 3.0 wird bei dieser Bedingung dagegen eine ReferenceError-Ausnahme ausgelöst.	Ausnahmen für nicht definierte Variable sind Hinweise auf mögliche Programmfehler und unterstützen Sie dabei, die Qualität von Anwendungen zu verbessern. Wenn Sie es jedoch bisher nicht gewohnt waren, Variablen zu initialisieren, müssen Sie aufgrund dieser neuen ActionScript-Funktionsweise Ihre Programmiergewohnheiten ändern.

Klassenname	Beschreibung	Hinweise
SecurityError	Eine SecurityError-Ausnahme wird ausgelöst, wenn eine Sicherheitsverletzung auftritt und der Zugriff verweigert wird.	Beispiele für solche Fehler: • Über die Begrenzung einer Sicherheits-Sandbox hinweg findet ein nicht autorisierter Zugriff auf eine Eigenschaft oder ein nicht autorisierter Aufruf einer Methode statt. • Es wurde versucht, auf eine URL-Adresse zuzugreifen, die von der Sicherheits-Sandbox nicht zugelassen ist. • Es wurde versucht, eine Socket-Verbindung mit einem Port herzustellen, aber die erforderliche Socket-Richtliniendatei war nicht vorhanden. • Es wurde versucht, auf die Kamera oder das Mikrofon des Benutzers zuzugreifen, aber der Benutzer hat den Zugriff auf das Gerät verweigert.
SyntaxError	Eine SyntaxError-Ausnahme wird ausgelöst, wenn ein Parsingfehler im ActionScript-Code auftritt.	SyntaxError-Ausnahmen werden in den folgenden Fällen ausgelöst: • ActionScript löst SyntaxError-Ausnahmen aus, wenn die RegExp-Klasse einen ungültigen regulären Ausdruck analysiert. • ActionScript löst SyntaxError-Ausnahmen aus, wenn die XMLDocument-Klasse ungültiges XML analysiert.
TypeError	Eine TypeError-Ausnahme wird ausgelöst, wenn sich der tatsächliche Typ eines Operanden vom erwarteten Typ unterscheidet.	TypeError-Ausnahmen werden in den folgenden Fällen ausgelöst: • Der Typ des tatsächlichen Parameters einer Funktion oder Methode konnte nicht automatisch in den formalen Parametertyp umgewandelt werden. • Einer Variablen wird ein Wert zugewiesen, der jedoch nicht automatisch in den Variablentyp umgewandelt werden kann. • Die rechte Seite des `is`- oder `instanceof`-Operators besitzt einen ungültigen Typ. • Das Schlüsselwort `super` wurde in unzulässiger Weise eingesetzt. • Das Nachschlagen einer Eigenschaft führt zu mehreren Bindungen und ist daher mehrdeutig. • Eine Methode wird für ein inkompatibles Objekt aufgerufen. So wird beispielsweise eine TypeError-Ausnahme ausgelöst, wenn eine Methode der RegExp-Klasse auf ein generisches Objekt übertragen und dann aufgerufen wird.
URIError	Eine URIError-Ausnahme wird ausgelöst, wenn eine der globalen Funktionen für die URI-Verarbeitung auf eine Weise eingesetzt wird, die mit ihrer Definition inkompatibel ist.	URIError-Ausnahmen werden in den folgenden Fällen ausgelöst: Für eine Funktion der Flash Player-API, z. B. `Socket.connect()`, die eine gültige URI erwartet, wurde eine ungültige URI angegeben.
VerifyError	Eine VerifyError-Ausnahme wird ausgelöst, wenn eine ungültige oder beschädigte SWF-Datei gefunden wird.	Wenn eine SWF-Datei eine andere SWF-Datei lädt, kann die übergeordnete SWF-Datei eine von der geladenen SWF-Datei hervorgerufene VerifyError-Ausnahme abfangen.

Fehlerklassen im flash.error-Paket

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das flash.error-Paket enthält Error-Klassen, die als Bestandteil der Flash-Laufzeitumgebungs-API angesehen werden. Im Gegensatz zu den beschriebenen Error-Klassen dienen die Klassen im flash.error-Paket zur Kennzeichnung von Fehlerereignissen, die für Flash-Laufzeitumgebungen spezifisch sind (wie Flash Player und Adobe AIR).

Klassenname	Beschreibung	Hinweise
EOFError	Eine EOFError-Ausnahme wird ausgelöst, wenn nach dem Ende der verfügbaren Daten ein Lesevorgang durchgeführt wird.	Beispielsweise wird eine EOFError-Ausnahme ausgelöst, wenn eine der Lesemethoden der IDataInput-Schnittstelle aufgerufen wird und für diese Anforderung nicht genug Daten vorhanden sind.
IllegalOperationError	Eine IllegalOperationError-Ausnahme wird ausgelöst, wenn eine Methode nicht implementiert ist oder wenn die Implementierung nicht die verwendeten Aufrufparameter unterstützt.	Beispiele für IllegalOperationError-Ausnahmen: • Eine Basisklasse wie DisplayObjectContainer bietet mehr Funktionalität als von der Bühne unterstützt wird. Wenn Sie beispielsweise versuchen, eine Maske auf der Bühne abzurufen oder festzulegen (mithilfe von `stage.mask`), löst die Flash-Laufzeitumgebung eine IllegalOperationError-Ausnahme mit der folgenden Meldung aus: „Die Stage-Klasse hat diese Eigenschaft oder Methode nicht implementiert". • Eine Unterklasse erbt eine Methode, die nicht erforderlich ist und nicht unterstützt wird. • Es werden bestimmte Eingabehilfen-Methoden aufgerufen, obwohl Flash Player ohne Eingabehilfen kompiliert wurde. • In einer Laufzeitversion von Flash Player werden Funktionen aufgerufen, die nur in der Authoring-Umgebung verfügbar sind. • Sie versuchen, den Namen eines Objekts festzulegen, das sich auf der Zeitleiste befindet.
IOError	Eine IOError-Ausnahme wird ausgelöst, wenn ein Ein- oder Ausgabefehler auftritt.	Beispielsweise wird diese Ausnahme ausgelöst, wenn ein Lese-/Schreibvorgang für eine nicht oder nicht mehr verfügbare Socketverbindung durchgeführt wird.

Klassenname	Beschreibung	Hinweise
MemoryError	Eine MemoryError-Ausnahme wird ausgelöst, wenn eine Speicherzuweisungsanforderung fehlschlägt.	In ActionScript Virtual Machine 2 ist standardmäßig keine Obergrenze für den Speicher vorgegeben, den ein ActionScript-Programm zuweist. Auf Desktopsystemen treten Speicherzuweisungsfehler nur selten auf. Diese Fehler werden ausgelöst, wenn das System den für einen Vorgang erforderlichen Speicher nicht zuweisen kann. Deshalb tritt diese Ausnahme auf Desktopsystemen nur selten auf, solange es sich nicht um sehr große Zuweisungsanforderungen handelt. Beispielsweise ist das Anfordern von 3.000.000.000 Byte nicht möglich, da 32-Bit-Microsoft® Windows®-Programme nur über einen Adressraum von 2 GB verfügen.
ScriptTimeoutError	Eine ScriptTimeoutError-Ausnahme wird ausgelöst, wenn das Skriptzeitlimit von 15 Sekunden erreicht ist. Indem Sie ScriptTimeoutError-Ausnahmen abfangen, können Sie besser auf die Überschreitung des Zeitlimits durch das Skript reagieren. Wenn keine Ausnahmeprozedur vorhanden ist, wird von der Ausnahmeprozedur für nicht abgefangene Ausnahmen ein Dialogfeld mit einer Fehlermeldung angezeigt.	Damit niemand diese Ausnahme in böswilliger Absicht abfängt und eine Endlosschleife hervorruft, kann nur die erste beim Ausführen eines bestimmten Skripts ausgelöste ScriptTimeoutError-Ausnahme abgefangen werden. Eine weitere ScriptTimeoutError-Ausnahme kann nicht mehr im Code abgefangen werden und wird sofort an die Ausnahmeprozedur für nicht abgefangene Ausnahmen weitergeleitet.
StackOverflowError	Eine StackOverflowError-Ausnahme wird ausgelöst, wenn der für das Skript verfügbare Stapelspeicher ausgeschöpft ist.	Eine StackOverflowError-Ausnahme weist möglicherweise darauf hin, dass eine unendliche Rekursion aufgetreten ist.

Beispiel für die Fehlerverarbeitung: CustomErrors-Anwendung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Anwendung „CustomErrors“ veranschaulicht Techniken für den Einsatz benutzerdefinierter Fehler beim Erstellen von Anwendungen. Dies sind im Einzelnen:

- Validieren eines XML-Pakets
- Programmieren eines benutzerdefinierten Fehlers
- Auslösen benutzerdefinierter Fehler
- Benachrichtigen von Benutzern beim Auslösen eines Fehlers

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „CustomErrors“ befinden sich im Ordner „Samples/CustomError“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
CustomErrors.mxml oder CustomErrors.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/errors/ApplicationError.as	Eine Klasse, die als Basisklasse für die beiden Fehlerklassen FatalError und WarningError dient.
com/example/programmingas3/errors/FatalError.as	Eine Klasse, die einen von der Anwendung ausgelösten FatalError-Fehler definiert. Diese Klasse erweitert die benutzerdefinierte ApplicationError-Klasse.
com/example/programmingas3/errors/Validator.as	Eine Klasse mit einer einzelnen Methode, mit der ein vom Benutzer bereitgestelltes XML-Paket mit Mitarbeiterdaten validiert wird.
com/example/programmingas3/errors/WarningError.as	Eine Klasse, die einen von der Anwendung ausgelösten WarningError-Fehler definiert. Diese Klasse erweitert die benutzerdefinierte ApplicationError-Klasse.

Überblick über die Anwendung „CustomErrors“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn die Anwendung geladen wird, wird die `initApp()`-Methode für Flex-Anwendungen aufgerufen oder der Zeitleistencode (ohne Funktion) wird für Flash Professional-Anwendungen ausgeführt. Dieser Code definiert ein XML-Beispielpaket, das mit der Validator-Klasse validiert wird. Folgender Code wird ausgeführt:

```
employeeXML =
    <employee id="12345">
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <costCenter>12345</costCenter>
        <costCenter>67890</costCenter>
    </employee>;
}
```

Das XML-Paket wird später in einer Instanz der TextArea-Komponente auf der Bühne angezeigt. Durch diesen Schritt können Sie das XML-Paket bearbeiten, bevor Sie es erneut validieren.

Wenn der Benutzer auf die Schaltfläche „Validate“ klickt, wird die `validateData()`-Methode aufgerufen. Diese Methode validiert das XML-Paket mit den Mitarbeiterdaten mithilfe der `validateEmployeeXML()`-Methode der Validator-Klasse. Im folgenden Code wird die `validateData()`-Methode veranschaulicht:

```
function validateData():void
{
    try
    {
        var tempXML:XML = XML(xmlText.text);
        Validator.validateEmployeeXML(tempXML);
        status.text = "The XML was successfully validated.";
    }
    catch (error:FatalError)
    {
        showFatalError(error);
    }
    catch (error:WarningError)
    {
        showWarningError(error);
    }
    catch (error:Error)
    {
        showGenericError(error);
    }
}
```

Zuerst wird mithilfe des Inhalts der TextArea-Komponenteninstanz `xmlText` ein temporäres XML-Objekt erstellt. Dann wird die `validateEmployeeXML()`-Methode in der benutzerdefinierten Validator-Klasse (com.example.programmingas3/errors/Validator.as) aufgerufen und das temporäre XML-Objekt wird als Parameter übergeben. Wenn das XML-Paket gültig ist, wird in der Label-Komponenteninstanz `status` eine Erfolgsmeldung angezeigt und die Anwendung beendet. Wenn die `validateEmployeeXML()`-Methode einen benutzerdefinierten Fehler auslöst (d. h. beim Auftreten von FatalError, WarningError oder eines generischen Error-Ereignisses), wird die entsprechende `catch` -Anweisung ausgeführt, die eine der Methoden `showFatalError()`, `showWarningError()` oder `showGenericError()` aufruft. Mit jeder dieser Methoden wird im Textbereich `statusTex`t eine entsprechende Meldung angezeigt, um den Benutzer über den genauen Fehler zu informieren. Alle diese Methoden aktualisieren auch die Label-Komponenteninstanz `status` mit einer speziellen Meldung.

Wie aus dem folgenden Codebeispiel hervorgeht, werden beim Auftreten eines schwerwiegenden Fehlers während des Validierens des XML-Pakets die Fehlermeldung im Textbereich `statusText` und die TextArea-Komponenteninstanz `xmlText` sowie die Button-Komponenteninstanz `validateBtn` deaktiviert:

```
function showFatalError(error:FatalError):void
{
    var message:String = error.message + "\n\n";
    var title:String = error.getTitle();
    statusText.text = message + " " + title + "\n\nThis application has ended.";
    this.xmlText.enabled = false;
    this.validateBtn.enabled = false;
    hideButtons();
}
```

Wenn anstelle eines schwerwiegenden Fehlers eine Warnung ausgelöst wird, wird die Fehlermeldung in der TextArea-Instanz `statusText` angezeigt, die TextField- und Button-Komponenteninstanzen `xmlText` werden jedoch nicht deaktiviert. Die `showWarningError()`-Methode zeigt die benutzerdefinierte Fehlermeldung im Textbereich `statusText` an. In der Meldung wird der Benutzer auch aufgefordert, zu entscheiden, ob das Validieren des XML-Pakets fortgesetzt oder das Skript abgebrochen werden soll. Im folgenden Codeauszug wird die `showWarningError()`-Methode veranschaulicht:

```
function showWarningError(error:WarningError):void
{
    var message:String = error.message + "\n\n" + "Do you want to exit this application?";
    showButtons();
    var title:String = error.getTitle();
    statusText.text = message;
}
```

Wenn der Benutzer auf „Ja“ oder „Nein“ klickt, wird die `closeHandler()`-Methode aufgerufen. Im folgenden Codeauszug wird die `closeHandler()`-Methode veranschaulicht:

```
function closeHandler(event:CloseEvent):void
{
    switch (event.detail)
    {
        case yesButton:
            showFatalError(new FatalError(9999));
            break;
        case noButton:
            statusText.text = "";
            hideButtons();
            break;
    }
}
```

Wenn sich der Benutzer dafür entscheidet, das Skript durch Klicken auf die Schaltfläche „Ja“ abzubrechen, wird eine FatalError-Ausnahme ausgelöst, die das Beenden der Anwendung zur Folge hat.

Erstellen einer benutzerdefinierten Validator-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die benutzerdefinierte Validator-Klasse enthält eine einzige Methode: `validateEmployeeXML()`. Die `validateEmployeeXML()`-Methode hat nur ein Argument, `employee`. Dies ist das zu validierende XML-Paket. Es folgt die `validateEmployeeXML()`-Methode:

```
public static function validateEmployeeXML(employee:XML):void
{
    // checks for the integrity of items in the XML
    if (employee.costCenter.length() < 1)
    {
        throw new FatalError(9000);
    }
    if (employee.costCenter.length() > 1)
    {
        throw new WarningError(9001);
    }
    if (employee.ssn.length() != 1)
    {
        throw new FatalError(9002);
    }
}
```

Für eine erfolgreiche Validierung muss ein Mitarbeiter einer (und nur einer) Kostenstelle angehören. Wenn der Mitarbeiter keiner Kostenstelle angehört, löst die Methode eine FatalError-Ausnahme aus, welches an die `validateData()`-Methode in der Datei der Hauptanwendung propagiert wird. Wenn der Mitarbeiter mehreren Kostenstellen angehört, wird eine WarningError-Ausnahme ausgelöst. Bei der letzten Überprüfung durch den XML-Validator wird sichergestellt, dass für den Mitarbeiter genau eine Sozialversicherungsnummer definiert ist (der `ssn`-Knoten des XML-Pakets). Wenn es nicht genau einen `ssn`-Knoten gibt, wird der FatalError-Fehler ausgelöst.

Sie können der `validateEmployeeXML()`-Methode zusätzliche Tests hinzufügen, beispielsweise um sicherzustellen, dass der `ssn`-Knoten eine gültige Nummer enthält oder dass für den Mitarbeiter mindestens eine Telefonnummer und E-Mail-Adresse definiert sind und beide Werte gültig sind. Sie können die XML-Daten auch dahingehend ändern, dass jeder Mitarbeiter über eine eindeutige ID verfügt und die ID des Vorgesetzten angegeben wird.

Definieren der ApplicationError-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die ApplicationError-Klasse dient als Basisklasse für die beiden Fehlerklassen FatalError und WarningError. Die ApplicationError-Klasse erweitert die Error-Klasse und definiert benutzerdefinierte Methoden und Eigenschaften. Dazu gehören eine Fehler-ID, der Schweregrad und ein XML-Objekt mit den Fehlercodes und Fehlermeldungen. Diese Klasse definiert auch zwei statische Konstanten, die zur Angabe des Schweregrads des jeweiligen Fehlertyps verwendet werden.

Es folgt die Konstruktormethode der ApplicationError-Klasse:

```
public function ApplicationError()
{
    messages =
        <errors>
            <error code="9000">
                <![CDATA[Employee must be assigned to a cost center.]]>
            </error>
            <error code="9001">
                <![CDATA[Employee must be assigned to only one cost center.]]>
            </error>
            <error code="9002">
                <![CDATA[Employee must have one and only one SSN.]]>
            </error>
            <error code="9999">
                <![CDATA[The application has been stopped.]]>
            </error>
        </errors>;
}
```

Jeder Fehlerknoten im XML-Objekt enthält einen eindeutigen numerischen Code und eine Fehlermeldung. Fehlermeldungen können mit E4X problemlos anhand des entsprechenden Fehlercodes ermittelt werden, wie mit der folgenden `getMessageText()`-Methode veranschaulicht wird:

```
public function getMessageText(id:int):String
{
    var message:XMLList = messages.error.(@code == id);
    return message[0].text();
}
```

Die `getMessageText()`-Methode erwartet das ganzzahlige Argument `id` und gibt einen String zurück. Das Argument `id` ist der Fehlercode des nachzuschlagenden Fehlers. Beispielsweise wird beim Übergeben von „9001“ für `id` die Fehlermeldung zurückgegeben, dass Mitarbeiter nur einer Kostenstelle zugeordnet sein dürfen. Wenn mehrere Fehler mit demselben Fehlercode vorhanden sind, wird nur die Fehlermeldung für das erste gefundene Ergebnis (`message[0]` im XMLList-Ergebnisobjekt) zurückgegeben.

Die nächste Methode dieser Klasse, `getTitle()`, erfordert keine Parameter und gibt einen Stringwert mit der Fehler-ID dieses spezifischen Fehlers zurück. Der Wert wird verwendet, damit Sie den genauen Fehler besser identifizieren können, der beim Validieren des XML-Pakets aufgetreten ist. Im folgenden Codeauszug wird die `getTitle()`-Methode veranschaulicht:

```
public function getTitle():String
{
    return "Error #" + id;
}
```

Die letzte Methode der ApplicationError-Klasse ist `toString()`. Diese Methode überschreibt die in der Error-Klasse definierte Funktion, damit Sie die Darstellung der Fehlermeldungen anpassen können. Die Methode gibt einen String zurück, der die jeweilige Fehlernummer und Fehlermeldung für den aufgetretenen Fehler enthält.

```
public override function toString():String
{
    return "[APPLICATION ERROR #" + id + "] " + message;
}
```

Definieren der FatalError-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die FatalError-Klasse erweitert die benutzerdefinierte ApplicationError-Klasse und definiert drei Methoden: den FatalError-Konstruktor, `getTitle()` und `toString()`. Die erste Methode, der FatalError-Konstruktor, erwartet als einziges Argument eine Ganzzahl, `errorID`. Der Schweregrad des Fehlers wird dann mithilfe der in der ApplicationError-Klasse definierten Konstanten festgelegt. Durch Aufrufen der `getMessageText()`-Methode der ApplicationError-Klasse wird die entsprechende Fehlermeldung ermittelt. Es folgt der FatalError-Konstruktor:

```
public function FatalError(errorID:int)
{
    id = errorID;
    severity = ApplicationError.FATAL;
    message = getMessageText(errorID);
}
```

Die nächste Methode der FatalError-Klasse, `getTitle()`, überschreibt die zuvor in der ApplicationError-Klasse definierte `getTitle()`-Methode und fügt dem Titel den Text „-- FATAL“ hinzu, um den Benutzer über das Auftreten eines schwerwiegenden Fehlers zu informieren. Die `getTitle()`-Methode wird wie folgt ausgeführt:

```
public override function getTitle():String
{
    return "Error #" + id + " -- FATAL";
}
```

Die letzte Methode dieser Klasse, `toString()`, überschreibt die in der ApplicationError-Klasse definierte `toString()`-Methode. Es folgt die `toString()`-Methode:

```
public override function toString():String
{
    return "[FATAL ERROR #" + id + "] " + message;
}
```

Definieren der WarningError-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die WarningError-Klasse erweitert die ApplicationError-Klasse und ist fast identisch mit der FatalError-Klasse. Wie aus dem folgenden Code ersichtlich ist, unterscheidet sie sich nur durch geringfügige Stringänderungen und das Setzen des Fehlerschweregrads auf ApplicationError.WARNING anstelle auf ApplicationError.FATAL:

```
public function WarningError(errorID:int)
{
    id = errorID;
    severity = ApplicationError.WARNING;
    message = super.getMessageText(errorID);
}
```

Kapitel 5: Verwenden von regulären Ausdrücken

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit einem regulären Ausdruck wird ein Muster beschrieben, das zum Suchen und Bearbeiten von übereinstimmendem Text in Strings verwendet wird. Reguläre Ausdrücke ähneln Strings, können jedoch spezielle Codes zum Beschreiben von Mustern und Wiederholungen enthalten. Der folgende reguläre Ausdruck entspricht beispielsweise einem String, der mit dem Zeichen A beginnt, gefolgt von einer oder mehreren sequenziellen Ziffern:

```
/A\d+/
```

In den folgenden Themen wird die grundlegende Syntax für den Aufbau von regulären Ausdrücken beschrieben. Reguläre Ausdrücke können jedoch vielschichtig sein und viele Nuancen aufweisen. Ausführliche Informationen zu regulären Ausdrücken finden Sie im Internet und in Buchhandlungen. Beachten Sie, dass reguläre Ausdrücke in verschiedenen Programmierumgebungen unterschiedlich implementiert werden. ActionScript 3.0 implementiert reguläre Ausdrücke entsprechend der Sprachspezifikation ECMAScript Version 3 (ECMA-262).

Verwandte Hilfethemen

RegExp

Grundlagen von regulären Ausdrücken

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein regulärer Ausdruck beschreibt ein Zeichenmuster. Mit regulären Ausdrücken wird in der Regel überprüft, ob ein Textwert einem bestimmten Muster entspricht (ob ein Benutzer z. B. eine Telefonnummer mit der richtigen Anzahl von Ziffern eingegeben hat). Mit regulären Ausdrücken werden normalerweise zudem Teile eines Textwerts ersetzt, die einem bestimmten Muster entsprechen.

Reguläre Ausdrücke können ganz einfach aufgebaut sein. Angenommen, Sie möchten prüfen, ob ein bestimmter String die Zeichenfolge „ABC“ enthält, oder alle Vorkommen von „ABC“ in einem String durch anderen Text ersetzen. In diesem Fall wird mit dem folgenden regulären Ausdruck ein Muster definiert, das aus den Buchstaben A, B und C in dieser Reihenfolge besteht:

```
/ABC/
```

Bei regulären Ausdrücken werden Schrägstriche (`/`) als Begrenzungszeichen verwendet.

Reguläre Ausdrücke können auch komplex und manchmal kryptisch sein, z. B. der folgende Ausdruck für eine gültige E-Mail-Adresse:

```
/([0-9a-zA-Z]+[-._+&])*[0-9a-zA-Z]+@([-0-9a-zA-Z]+[.])+[a-zA-Z]{2,6}/
```

In der Regel verwenden Sie reguläre Ausdrücke, um in Strings nach Mustern zu suchen und um Zeichen zu ersetzen. In diesen Fällen erstellen Sie ein Objekt für den jeweiligen regulären Ausdruck und verwenden es als Parameter für eine der verschiedenen Methoden der String-Klasse. Bei den folgenden Methoden der String-Klasse können reguläre Ausdrücke als Parameter verwendet werden: `match()`, `replace()`, `search()` und `split()`. Weitere Informationen zu diesen Methoden finden Sie unter „Suchen von Mustern in Strings und Ersetzen von Teilstrings“ auf Seite 17.

Die RegExp-Klasse umfasst folgende Methoden: `test()` und `exec()`. Weitere Informationen finden Sie unter „Methoden für das Verwenden regulärer Ausdrücke bei Strings“ auf Seite 96.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die mit dieser Funktion zu tun haben:

Escape-Zeichen Ein Zeichen, das angibt, dass das folgende Zeichen nicht als Literalwert, sondern als Metazeichen behandelt werden soll. In der Syntax für reguläre Ausdrücke ist der umgekehrte Schrägstrich (\) das Escape-Zeichen. Ein umgekehrter Schrägstrich gefolgt von einem weiteren Zeichen steht deshalb nicht für das Zeichen selbst, sondern ist ein Sondercode.

Flag Ein Zeichen, das eine Option zur Verwendung eines Musters für reguläre Ausdrücke angibt, beispielsweise ob zwischen Groß- und Kleinschreibung unterschieden wird.

Metazeichen Ein Zeichen mit besonderer Bedeutung in einem Muster für reguläre Ausdrücke. Es steht im Muster nicht für das Zeichen selbst.

Quantifizierer Ein (oder mehrere) Zeichen zum Angeben, wie oft ein Teil des Musters sich wiederholen soll. Beispielsweise kann mit einem Quantifizierer angegeben werden, dass eine US-Postleitzahl aus fünf oder neun Ziffern bestehen muss.

Regulärer Ausdruck Eine Programmanweisung zum Definieren eines Zeichenmusters, mit dem andere Strings auf Übereinstimmung mit diesem Muster überprüft oder Teile eines Strings ersetzt werden können.

Syntax regulärer Ausdrücke

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In diesem Abschnitt werden alle Elemente der Syntax für reguläre Ausdrücke in ActionScript erläutert. Reguläre Ausdrücke können vielschichtig sein und viele Nuancen aufweisen. Ausführliche Informationen zu regulären Ausdrücken finden Sie im Internet und in Buchhandlungen. Beachten Sie, dass reguläre Ausdrücke in verschiedenen Programmierumgebungen unterschiedlich implementiert werden. ActionScript 3.0 implementiert reguläre Ausdrücke entsprechend der Sprachspezifikation ECMAScript Version 3 (ECMA-262).

Im Allgemeinen werden reguläre Ausdrücke für Muster verwendet, die komplizierter sind als einfache Zeichenstrings. Mit dem folgenden regulären Ausdruck wird beispielsweise ein Muster definiert, das aus den Buchstaben A, B und C sowie einer beliebigen Ziffer besteht:

```
/ABC\d/
```

Mit dem Code `\d` wird eine beliebige Ziffer angegeben. Der umgekehrte Schrägstrich (`\`), das sogenannte Escape-Zeichen, hat kombiniert mit dem Zeichen, das nach dem umgekehrten Schrägstrich folgt (in diesem Fall der Buchstabe d), in regulären Ausdrücken eine besondere Bedeutung.

Mit dem folgenden regulären Ausdruck wird das Muster der Buchstaben ABC, gefolgt von einer beliebigen Anzahl von Ziffern definiert (beachten Sie das Sternchen):

```
/ABC\d*/
```

Das Sternchen (`*`) ist ein *Metazeichen*. Ein Metazeichen ist ein Zeichen, das in regulären Ausdrücken eine besondere Bedeutung hat. Das Sternchen ist ein bestimmter Metazeichentyp, der als *Quantifizierer* bezeichnet wird und mit dem die Anzahl der Wiederholungen eines Zeichens oder einer Zeichengruppe quantifiziert wird. Weitere Informationen finden Sie unter „Quantifizierer“ auf Seite 88.

Ein regulärer Ausdruck kann neben Mustern auch Flags enthalten, mit denen angegeben wird, auf welche Weise Entsprechungen des regulären Ausdrucks ermittelt werden. Im folgenden Ausdruck wird beispielsweise das `i`-Flag verwendet, mit dem angegeben wird, dass die Groß- und Kleinschreibung in übereinstimmenden Strings ignoriert wird:

```
/ABC\d*/i
```

Weitere Informationen finden Sie unter „Flags und Eigenschaften“ auf Seite 92.

Sie können reguläre Ausdrücke mit den folgenden Methoden der String-Klasse verwenden: `match()`, `replace()` und `search()`. Weitere Informationen zu diesen Methoden finden Sie unter „Suchen von Mustern in Strings und Ersetzen von Teilstrings“ auf Seite 17.

Erstellen einer Instanz eines regulären Ausdrucks

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine Instanz eines regulären Ausdrucks kann auf zwei verschiedene Weisen erstellt werden. Entweder wird ein Schrägstrich (`/`) als Trennzeichen in einem regulären Ausdruck verwendet, oder der `new`-Konstruktor. Die folgenden regulären Ausdrücke sind beispielsweise gleichwertig:

```
var pattern1:RegExp = /bob/i;
var pattern2:RegExp = new RegExp("bob", "i");
```

Schrägstriche werden auf die gleiche Weise als Trennzeichen in regulären Ausdrucksliteralen verwendet wie Anführungszeichen als Trennzeichen in Stringliteralen. Mit dem Teil eines regulären Ausdrucks innerhalb der Schrägstriche wird das *Muster definiert.* Ein regulärer Ausdruck kann zudem nach dem letzten trennenden Schrägstrich *Flags* enthalten. Diese Flags sind Bestandteil des regulären Ausdrucks, jedoch vom entsprechenden Muster getrennt.

Bei Verwendung des `new`-Konstruktors wird ein regulärer Ausdruck mithilfe von zwei Strings definiert. Mit dem ersten String wird das Muster festgelegt. Mit dem zweiten String werden die Flags definiert, wie im folgenden Beispiel dargestellt:

```
var pattern2:RegExp = new RegExp("bob", "i");
```

Wenn Sie einen Schrägstrich *innerhalb* eines regulären Ausdrucks verwenden möchten, bei dem Schrägstriche als Trennzeichen dienen, müssen Sie dem Schrägstrich einen umgekehrten Schrägstrich (`\`) als Escape-Zeichen voranstellen. Der folgende reguläre Ausdruck entspricht beispielsweise dem Muster `1/2`:

```
var pattern:RegExp = /1\/2/;
```

Wenn Sie Anführungszeichen *innerhalb* eines regulären Ausdrucks verwenden möchten, der durch den `new`-Konstruktor definiert wird, müssen Sie vor den Anführungszeichen einen umgekehrten Schrägstrich (`\`) als Escape-Zeichen einfügen (wie beim Definieren von Stringliteralen). Der folgende reguläre Ausdruck entspricht beispielsweise dem Muster `eat at "joe's"`:

```
var pattern1:RegExp = new RegExp("eat at \"joe's\"", "");
var pattern2:RegExp = new RegExp('eat at "joe\'s"', "");
```

Fügen Sie in regulären Ausdrücken, bei denen Schrägstriche als Trennzeichen verwendet werden, keine Anführungszeichen mit umgekehrten Schrägstrichen als Escape-Zeichen ein. Verwenden Sie ebenso keine Schrägstriche mit Escape-Zeichen in regulären Ausdrücken, die durch den `new`-Konstruktor definiert werden. Die folgenden Ausdrücke sind gleichwertig und definieren das Muster `1/2 "joe's"`:

```
var pattern1:RegExp = /1\/2 "joe's"/;
var pattern2:RegExp = new RegExp("1/2 \"joe's\"", "");
var pattern3:RegExp = new RegExp('1/2 "joe\'s"', '');
```

Geben Sie den umgekehrten Schrägstrich zweimal ein, wenn Sie in einem regulären Ausdruck, der durch den `new`-Konstruktor definiert wird, eine Metasequenz verwenden möchten, die mit einem umgekehrten Schrägstrich (`\`) beginnt, z. B. `\d` (entspricht einer Ziffer):

```
var pattern:RegExp = new RegExp("\\d+", ""); // matches one or more digits
```

In diesem Fall müssen Sie den umgekehrten Schrägstrich zweimal eingeben, da der erste Parameter der `RegExp()`-Konstruktormethode ein String ist und in einem Stringliteral ein umgekehrter Schrägstrich zweimal eingegeben werden muss, damit er als einfacher umgekehrter Schrägstrich erkannt wird.

In den folgenden Abschnitten wird die Syntax zum Definieren von regulären Ausdrücken beschrieben.

Weitere Informationen zu Flags finden Sie unter „Flags und Eigenschaften“ auf Seite 92.

Zeichen, Metazeichen und Metasequenzen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der einfachste reguläre Ausdruck entspricht einer Sequenz von Zeichen, wie im folgenden Beispiel dargestellt:

```
var pattern:RegExp = /hello/;
```

Die folgenden Zeichen, die als Metazeichen bezeichnet werden, haben in regulären Ausdrücken eine besondere Bedeutung:

```
^ $ \ . * + ? ( ) [ ] { } |
```

Der nachstehende reguläre Ausdruck entspricht beispielsweise dem folgenden Muster: Buchstabe A, gefolgt von keiner oder mindestens einer Instanz des Buchstabens B (diese Wiederholung wird durch das Sternchen-Metazeichen angegeben), gefolgt vom Buchstaben C:

```
/AB*C/
```

Wenn Sie ein Metazeichen ohne die entsprechende besondere Bedeutung in einem regulären Ausdruck einfügen möchten, müssen Sie den umgekehrten Schrägstrich (`\`) als Escape-Zeichen voranstellen. Der nachstehende reguläre Ausdruck entspricht beispielsweise dem folgenden Muster: Buchstabe A, Buchstabe B, Sternchen, Buchstabe C:

```
var pattern:RegExp = /AB\*C/;
```

Wie Metazeichen verfügen auch *Metasequenzen* in regulären Ausdrücken über eine besondere Bedeutung. Eine Metasequenz umfasst mehrere Zeichen. In den folgenden Abschnitten finden Sie ausführliche Informationen zur Verwendung von Metazeichen und Metasequenzen.

Metazeichen

In der folgenden Tabelle sind die Metazeichen aufgeführt, die Sie in regulären Ausdrücken verwenden können:

Metazeichen	Beschreibung
^ (Caretzeichen)	Entspricht dem Beginn des Strings. Mit dem gesetzten `m`-Flag (`multiline`) entspricht das Caretzeichen auch dem Anfang einer Zeile (siehe „Flags und Eigenschaften“ auf Seite 92). Wenn das Caretzeichen am Anfang einer Zeichenklasse verwendet wird, gibt es eine Negation und nicht den Beginn eines Strings an. Weitere Informationen finden Sie unter „Zeichenklassen“ auf Seite 86.
$ (Dollarzeichen)	Entspricht dem Ende des Strings. Mit dem gesetzten `m`-Flag (`multiline`) entspricht `$` auch der Position vor einem Zeilenvorschub (`\n`). Weitere Informationen finden Sie unter „Flags und Eigenschaften“ auf Seite 92.
\ (umgekehrter Schrägstrich)	Dient als Escape-Zeichen für die besondere Metazeichenbedeutung von Sonderzeichen. Verwenden Sie den umgekehrten Schrägstrich auch, wenn Sie einen normalen Schrägstrich in einem regulären Ausdrucksliteral verwenden möchten, z. B. in `/1\/2/` (um folgendes Muster anzugeben: Zeichen 1, Schrägstrich, Zeichen 2).
. (Punkt)	Entspricht einem einzelnen Zeichen. Ein Punkt entspricht nur dann einem Zeilenvorschubzeichen (`\n`), wenn das `s`-Flag (`dotall`) gesetzt ist. Weitere Informationen finden Sie unter „Flags und Eigenschaften“ auf Seite 92.
* (Sternchen)	Entspricht dem vorherigen Element, das nicht, einmal oder mehrmals wiederholt wird. Weitere Informationen finden Sie unter „Quantifizierer“ auf Seite 88.
+ (Pluszeichen)	Entspricht dem vorherigen Element, das mindestens einmal wiederholt wird. Weitere Informationen finden Sie unter „Quantifizierer“ auf Seite 88.
? (Fragezeichen)	Entspricht dem vorherigen Element, das nicht oder einmal wiederholt wird. Weitere Informationen finden Sie unter „Quantifizierer“ auf Seite 88.
(und)	Definiert Gruppen innerhalb eines regulären Ausdrucks. Gruppen können für Folgendes verwendet werden: • Einschränken des Gültigkeitsbereichs des |-Auswahlzeichens: `/(a|b|c)d/` • Definieren des Gültigkeitsbereichs eines Quantifizierers: `/(walla.){1,2}/` • In Rückverweisen. `\1` im folgenden regulären Ausdruck entspricht beispielsweise allen Zeichen, die der ersten in Klammern eingeschlossenen Gruppe des Musters entsprechen: • `/(\w*) wird wiederholt: \1/` Weitere Informationen finden Sie unter „Gruppen“ auf Seite 90.
[und]	Definiert eine Zeichenklasse, mit der mögliche Entsprechungen für ein einzelnes Zeichen festgelegt werden: `/[aeiou]/` entspricht jedem der angegebenen Zeichen. Verwenden Sie in Zeichenklassen einen Bindestrich (`-`), um einen Bereich von Zeichen anzugeben: `/[A-Z0-9]/` entspricht A bis Z in Großschreibung oder 0 bis 9. Fügen Sie in Zeichenklassen einen umgekehrten Schrägstrich als Escape-Zeichen für das Zeichen „]“ und das Zeichen „-“ ein: `/[+\-]\d+/` entspricht entweder `+` oder `-` vor mindestens einer Ziffer. In Zeichenklassen werden Zeichen, die normalerweise Metazeichen sind, als normale Zeichen (und nicht als Metazeichen) behandelt, ohne dass ein umgekehrter Schrägstrich eingefügt werden muss: `/[$]/£` entspricht entweder `$oder` £. Weitere Informationen finden Sie unter „Zeichenklassen“ auf Seite 86.
| *(senkrechter Strich)*	Wird zur Auswahl aus mehreren Alternativen verwendet. Entspricht entweder dem Teil links vom Strich oder dem Teil rechts vom Strich: `/abc|xyz/` entspricht entweder `abc` oder `xyz`.

Metasequenzen

Metasequenzen sind Zeichensequenzen, die in regulären Ausdrücken eine besondere Bedeutung haben. In der folgenden Tabelle sind diese Metasequenzen beschrieben:

Metasequenz	Beschreibung

`{n}`
`{n,}`
und
`{n,n}`
	Gibt einen numerischen Quantifizierer oder Quantifiziererbereich für das vorherige Element an:
	`/A{27}/` entspricht dem Zeichen `A`, das `27` Mal wiederholt wird.
	`/A{3,}/` entspricht dem Zeichen `A`, das mindestens `3` Mal wiederholt wird.
	`/A{3,5}/` entspricht dem Zeichen `A`, das zwischen `3` und `5` Mal wiederholt wird.
	Weitere Informationen finden Sie unter „Quantifizierer“ auf Seite 88.

`\b`	Entspricht der Position zwischen einem Wortzeichen und einem Nichtwortzeichen. Entspricht auch dem Beginn oder Ende eines Strings, wenn das erste oder letzte Zeichen im String ein Wortzeichen ist.
`\B`	Entspricht der Position zwischen zwei Wortzeichen. Entspricht auch der Position zwischen zwei Nichtwortzeichen.
`\d`	Entspricht einer Dezimalziffer.
`\D`	Entspricht jedem Zeichen, das keine Ziffer ist.
`\f`	Entspricht einem Seitenvorschubzeichen.
`\n`	Entspricht dem Zeilenvorschubzeichen.
`\r`	Entspricht dem Wagenrücklaufzeichen.
`\s`	Entspricht einem beliebigen Leerraumzeichen (Leerzeichen, Tabulator, Zeilenvorschub oder Wagenrücklauf).
`\S`	Entspricht jedem Zeichen, das kein Leerraumzeichen ist.
`\t`	Entspricht dem Tabulatorzeichen.
`\unnnn`	Entspricht dem Unicode-Zeichen mit dem durch die Hexadezimalzahl *nnnn* angegebenen Zeichencode. Beispielsweise steht `\u263a` für das Smiley-Zeichen.
`\v`	Entspricht einem Zeichen für den vertikalen Vorschub.
`\w`	Entspricht einem Wortzeichen (`AZ`–, `az`–, `0-9` oder _). Beachten Sie, dass `\w` keinen Zeichen entspricht, die im Englischen nicht vorkommen, z. B. `ä`, `ñ` oder `ç`.
`\W`	Entspricht jedem Zeichen, das kein Wortzeichen ist.
`\\xnn`	Entspricht dem Zeichen mit dem angegebenen ASCII-Wert, der durch die Hexadezimalzahl *nn* definiert ist.

Zeichenklassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit Zeichenklassen können Sie eine Liste von Zeichen angeben, die einer Position in einem regulären Ausdruck entsprechen. Zeichenklassen werden durch eckige Klammern (`[` und `]`) definiert. Mit dem folgenden regulären Ausdruck wird beispielsweise eine Zeichenklasse definiert, die `bag`, `beg`, `big`, `bog` oder `bug` entspricht:

```
/b[aeiou]g/
```

Escape-Sequenzen in Zeichenklassen

Die meisten Metazeichen und Metasequenzen mit einer besonderen Bedeutung in regulären Ausdrücken haben diese Bedeutung *nicht* in Zeichenklassen. In einem regulären Ausdruck wird das Sternchen beispielsweise für eine Wiederholung verwendet. Dies ist nicht der Fall, wenn das Sternchen in einer Zeichenklasse verwendet wird. Die folgende Zeichenklasse entspricht tatsächlich dem Sternchen sowie allen anderen aufgeführten Zeichen:

```
/[abc*123]/
```

Die drei in der folgenden Tabelle aufgeführten Zeichen werden in Zeichenklassen jedoch als Metazeichen mit einer besonderen Bedeutung verwendet:

Metazeichen	Bedeutung in Zeichenklassen
`]`	Definiert das Ende der Zeichenklasse.
`-`	Definiert einen Zeichenbereich (weiter Informationen finden Sie im folgenden Abschnitt „Zeichenbereiche in Zeichenklassen").
`\`	Definiert Metasequenzen und hebt die besondere Bedeutung von Metazeichen auf.

Damit diese Zeichen als normale Zeichen erkannt werden (ohne besondere Metazeichenbedeutung), muss vor dem entsprechenden Zeichen ein umgekehrter Schrägstrich als Escape-Zeichen eingefügt werden. Der folgende reguläre Ausdruck enthält beispielsweise eine Zeichenklasse, die einem der vier Symbole (`$`, `\`, `]` oder `-`) entspricht:

```
/[$\\\]\-]/
```

Neben den Metazeichen, die ihre besondere Bedeutung beibehalten, werden die folgenden Metasequenzen in Zeichenklassen als Metasequenzen verwendet:

Metasequenz	Bedeutung in Zeichenklassen
`\n`	Entspricht einem Zeilenvorschubzeichen.
`\r`	Entspricht einem Wagenrücklaufzeichen.
`\t`	Entspricht einem Tabulatorzeichen.
`\u`***nnnn***	Entspricht dem Zeichen mit dem angegebenen Unicode-Codepunktwert (definiert durch die Hexadezimalzahl *nnnn*).
`\\x`***nn***	Entspricht dem Zeichen mit dem angegebenen ASCII-Wert (definiert durch die Hexadezimalzahl *nn*).

Die anderen Metasequenzen und Metazeichen in regulären Ausdrücken werden innerhalb von Zeichenklassen wie normale Zeichen behandelt.

Zeichenbereiche in Zeichenklassen

Mit einem Bindestrich können Sie einen Bereich von Zeichen angeben, z. B. `A-Z`, `a-z` oder `0-9`. Diese Zeichen müssen einen gültigen Zeichenbereich im Zeichensatz bilden. Die folgende Zeichenklasse entspricht beispielsweise allen Zeichen im Bereich `a-z` oder allen Ziffern:

```
/[a-z0-9]/
```

Mit dem ASCII-Zeichencode `\\x`*nn* können Sie zudem einen Bereich durch ASCII-Werte angeben. Die folgende Zeichenklasse entspricht beispielsweise allen Zeichen im Zeichensatz der ASCII-Sonderzeichen (z. B. `ä` oder `ß`):

```
\\x
```

Negierte Zeichenklassen

Wenn Sie ein Caretzeichen (^) am Anfang einer Zeichenklasse verwenden, wird diese Klasse negiert, d. h., alle nicht aufgeführten Zeichen werden als Entsprechungen erkannt. Die folgende Zeichenklasse entspricht allen Zeichen, *mit Ausnahme* von kleingeschriebenen Buchstaben (a–z–) und Ziffern:

```
/[^a-z0-9]/
```

Sie müssen das Caretzeichen (^) am *Anfang* einer Zeichenklasse eingeben, um eine Negation anzugeben. Andernfalls wird das Caretzeichen einfach zu den Zeichen in der Zeichenklasse hinzugefügt. Die folgende Zeichenklasse entspricht beispielsweise einem Zeichen in einem Bereich mit bestimmten Symbolzeichen, einschließlich des Caretzeichens:

```
/[!.,#+*%$&^]/
```

Quantifizierer

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit Quantifizierern können Sie Wiederholungen von Zeichen oder Sequenzen in Mustern wie folgt angeben:

Quantifizierer-Metazeichen	Beschreibung
* (Sternchen)	Entspricht dem vorherigen Element, das nicht, einmal oder mehrmals wiederholt wird.
+ (Pluszeichen)	Entspricht dem vorherigen Element, das mindestens einmal wiederholt wird.
? (Fragezeichen)	Entspricht dem vorherigen Element, das nicht oder einmal wiederholt wird.
{*n*} {*n*,} und {*n*,*n*}	Gibt einen numerischen Quantifizierer oder Quantifiziererbereich für das vorherige Element an: /A{27}/ entspricht dem Zeichen A, das 27 Mal wiederholt wird. /A{3,}/ entspricht dem Zeichen A, das mindestens 3 Mal wiederholt wird. /A{3,5}/ entspricht dem Zeichen A, das zwischen 3 und 5 Mal wiederholt wird.

Sie können einen Quantifizierer auf ein einzelnes Zeichen, auf eine Zeichenklasse oder auf eine Gruppe anwenden:

- /a+/ entspricht dem Zeichen a, das mindestens ein Mal wiederholt wird.
- /\d+/ entspricht mindestens einer Ziffer.
- /[abc]+/ entspricht einer Wiederholung mindestens eines Zeichens, wobei es sich jeweils um das Zeichen a, b oder c handelt.
- /(sehr,)*/ entspricht dem Wort sehr, gefolgt von einem Komma und einem Leerzeichen, das nicht, einmal oder mehrmals wiederholt wird.

Sie können Quantifizierer innerhalb von in Klammern eingeschlossenen Gruppen verwenden, auf die Quantifizierer angewendet werden. Der folgende Quantifizierer entspricht beispielsweise Strings wie Wort und Wort-Wort-Wort:

```
/\w+(-\w+)*/
```

In der Standardeinstellung wird mit regulären Ausdrücken eine sogenannte *gierige Suche durchgeführt.* Für alle Teilmuster in einem regulären Ausdruck (z. B. .*) wird im String nach möglichst vielen übereinstimmenden Zeichen gesucht, bevor der nächste Teil des regulären Ausdrucks verarbeitet wird. Betrachten Sie beispielsweise den folgenden regulären Ausdruck und String:

```
var pattern:RegExp = /<p>.*<\/p>/;
str:String = "<p>Paragraph 1</p> <p>Paragraph 2</p>";
```

Der reguläre Ausdruck stimmt mit dem gesamten String überein:

```
<p>Paragraph 1</p> <p>Paragraph 2</p>
```

Wenn Sie jedoch beispielsweise Übereinstimmungen in nur einer `<p>...</p>`-Gruppe suchen, erreichen Sie dies wie folgt:

```
<p>Paragraph 1</p>
```

Fügen Sie ein Fragezeichen (`?`) nach einem Quantifizierer ein, um ihn in einen sogenannten *genügsamen Quantifizierer* zu ändern. Der folgende reguläre Ausdruck mit dem genügsamen Quantifizierer `*?` entspricht beispielsweise `<p>`, gefolgt von der Mindestanzahl der möglichen Zeichen (daher „genügsam"), gefolgt von `</p>`:

```
/<p>.*?<\/p>/
```

Berücksichtigen Sie dabei die folgenden Punkte im Hinblick auf Quantifizierer:

- Mit den Quantifizierern `{0}` und `{0,0}` wird ein Element nicht aus einer Übereinstimmung ausgeschlossen.
- Kombinieren Sie nicht mehrere Quantifizierer miteinander, z. B. `/abc+*/`.
- Der Punkt (.) umfasst mehrere Zeilen nur, wenn das `s`-Flag (`dotall`) gesetzt ist, auch wenn nach dem Punkt ein `*`-Quantifizierer eingefügt wird. Betrachten Sie den folgenden Beispielcode:

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*<\/p>/;
trace(str.match(re)); // null;

re = /<p>.*<\/p>/s;
trace(str.match(re));
    // output: <p>Test
    //                 Multiline</p>
```

Weitere Informationen finden Sie unter „Flags und Eigenschaften" auf Seite 92.

Auswahl aus mehreren Alternativen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei Verwendung des `|`-Zeichens (senkrechter Strich) in einem regulären Ausdruck wird nach alternativen Übereinstimmungen gesucht. Der folgende reguläre Ausdruck entspricht beispielsweise einem der Wörter `cat, dog, pig, rat`:

```
var pattern:RegExp = /cat|dog|pig|rat/;
```

Mit Klammern können Sie Gruppen festlegen, um den Bereich des |-Auswahlzeichens zu beschränken. Der folgende reguläre Ausdruck entspricht `cat`, gefolgt von `nap` oder `nip`:

```
var pattern:RegExp = /cat(nap|nip)/;
```

Weitere Informationen finden Sie unter „Gruppen" auf Seite 90.

Die folgenden beiden regulären Ausdrücke, der eine mit dem `|`-Auswahlzeichen und der andere mit einer Zeichenklasse (definiert durch `[` und `]`), sind gleichwertig:

```
/1|3|5|7|9/
/[13579]/
```

Weitere Informationen finden Sie unter „Zeichenklassen" auf Seite 86.

Gruppen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können eine Gruppe in einem regulären Ausdruck durch Klammern angeben, wie im Folgenden dargestellt:

```
/class-(\d*)/
```

Eine Gruppe ist ein Teilbereich eines Musters. Gruppen können für Folgendes verwendet werden:

- Anwenden eines Quantifizierers auf mehrere Zeichen
- Trennen von Teilmustern zur alternativen Auswahl (mit dem Zeichen `|`)
- Erfassen übereinstimmender Teilstrings (z. B. Verwenden von `\1` in einem regulären Ausdruck, um einer vorherigen übereinstimmenden Gruppe zu entsprechen, oder analoges Verwenden von `$1` in der `replace()`-Methode der String-Klasse)

In den folgenden Abschnitten finden Sie ausführliche Informationen zur Verwendung von Gruppen.

Verwenden von Gruppen mit Quantifizierern

Wenn Sie keine Gruppe verwenden, werden Quantifizierer wie folgt auf Zeichen oder Zeichenklassen angewendet, die vor dem Quantifizierer stehen:

```
var pattern:RegExp = /ab*/ ;
// matches the character a followed by
// zero or more occurrences of the character b

pattern = /a\d+/;
// matches the character a followed by
// one or more digits

pattern = /a[123]{1,3}/;
// matches the character a followed by
// one to three occurrences of either 1, 2, or 3
```

Sie können jedoch eine Gruppe verwenden, um einen Quantifizierer auf mehrere Zeichen oder Zeichenklassen anzuwenden:

```
var pattern:RegExp = /(ab)*/;
// matches zero or more occurrences of the character a
// followed by the character b, such as ababab

pattern = /(a\d)+/;
// matches one or more occurrences of the character a followed by
// a digit, such as a1a5a8a3

pattern = /(spam ){1,3}/;
// matches 1 to 3 occurrences of the word spam followed by a space
```

Weitere Informationen zu Quantifizierern finden Sie unter „Quantifizierer“ auf Seite 88.

Verwenden von Gruppen mit dem Auswahlzeichen (|)

Mithilfe von Gruppen können Sie eine Zeichengruppe definieren, auf die das Auswahlzeichen (`|`) angewendet werden soll, wie im Folgenden dargestellt:

```
var pattern:RegExp = /cat|dog/;
// matches cat or dog

pattern = /ca(t|d)og/;
// matches catog or cadog
```

Verwenden von Gruppen zum Zwischenspeichern übereinstimmender Teilstrings

Wenn Sie in einem Muster eine Standardgruppe in Klammern definieren, können Sie später im regulären Ausdruck auf die Gruppe verweisen. Dies wird als *Rückverweis* bezeichnet. Die entsprechenden Gruppen stellen *zwischengespeicherte Gruppen* dar. Die Sequenz `\1` im folgenden regulären Ausdruck entspricht beispielsweise allen Teilstrings, die der in Klammern eingeschlossenen zwischengespeicherten Gruppe entsprechen:

```
var pattern:RegExp = /(\d+)-by-\1/;
// matches the following: 48-by-48
```

Sie können bis zu 99 dieser Rückverweise in einem regulären Ausdruck angeben, indem Sie `\1`, `\2`, ... , `\99` eingeben.

Ebenso können Sie bei der `replace()`-Methode der String-Klasse `$1$99` verwenden, um die mit der zwischengespeicherten Gruppe übereinstimmenden Teilstrings im Ersetzungsstring einzufügen:

```
var pattern:RegExp = /Hi, (\w+)\./;
var str:String = "Hi, Bob.";
trace(str.replace(pattern, "$1, hello."));
    // output: Bob, hello.
```

Bei Verwendung von zwischengespeicherten Gruppen geben die `exec()`-Methode der RegExp-Klasse und die `match()`-Methode der String-Klasse Teilstrings zurück, die diesen Gruppen entsprechen:

```
var pattern:RegExp = /(\w+)@(\w+).(\w+)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
    // bob@example.com,bob,example,com
```

Verwenden von nicht zwischengespeicherten Gruppen und Vorschaugruppen

Eine nicht zwischengespeicherte Gruppe wird nur zum Gruppieren verwendet. Sie wird nicht zwischengespeichert und kann deshalb nicht mit nummerierten Rückverweisen referenziert werden. Mithilfe von `(?:` und `)` können Sie nicht zwischengespeicherte Gruppen wie folgt definieren:

```
var pattern = /(?:com|org|net);
```

Beachten Sie beispielsweise den Unterschied zwischen dem Einfügen von `(com|org)` in einer zwischengespeicherten und in einer nicht zwischengespeicherten Gruppe (mit der `exec()`-Methode werden zwischengespeicherte Gruppen nach der abgeschlossenen Suche aufgelistet):

```
var pattern:RegExp = /(\w+)@(\w+).(com|org)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
// bob@example.com,bob,example,com

//noncapturing:
var pattern:RegExp = /(\w+)@(\w+).(?:com|org)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
    // bob@example.com,bob,example
```

Ein spezieller Typ der zwischengespeicherten Gruppe ist die *Vorschaugruppe*, von der es zwei Typen gibt: die *positive Vorschaugruppe* und die *negative Vorschaugruppe*.

Mithilfe von `(?=` und `)` können Sie eine positive Vorschaugruppe definieren, die nur das Teilmuster ab der übereinstimmenden Position enthält. Der Teil des Strings, der der positiven Vorschaugruppe entspricht, kann jedoch auch noch mit weiteren Mustern im regulären Ausdruck übereinstimmen. Da `(?=e)` im folgenden Code eine positive Vorschaugruppe ist, kann das mit ihr übereinstimmende Zeichen `e` auch einem darauf folgenden Teil im regulären Ausdruck entsprechen, in diesem Fall der zwischengespeicherten Gruppe `(\w*)`:

```
var pattern:RegExp = /sh(?=e)(\w*)/i;
var str:String = "Shelly sells seashells by the seashore";
trace(pattern.exec(str));
// Shelly,elly
```

Mithilfe von `(?!` und `)` können Sie eine negative Vorschaugruppe definieren, mit der angegeben wird, dass das Teilmuster in der Gruppe *nicht* an der entsprechenden Position übereinstimmen darf. Zum Beispiel:

```
var pattern:RegExp = /sh(?!e)(\w*)/i;
var str:String = "She sells seashells by the seashore";
trace(pattern.exec(str));
// shore,ore
```

Verwenden von benannten Gruppen

Eine benannte Gruppe ist ein Gruppentyp in einem regulären Ausdruck, für die ein benannter Bezeichner angegeben ist. Mithilfe von `(?P<name>` und `)` können Sie eine benannte Gruppe definieren. Der folgende reguläre Ausdruck enthält beispielsweise eine benannte Gruppe mit dem Bezeichner `digits`:

```
var pattern = /[a-z]+(?P<digits>\d+)[a-z]+/;
```

Bei Verwendung der `exec()`-Methode wird eine übereinstimmende benannte Gruppe als Eigenschaft des `result`-Arrays hinzugefügt:

```
var myPattern:RegExp = /([a-z]+)(?P<digits>\d+)[a-z]+/;
var str:String = "a123bcd";
var result:Array = myPattern.exec(str);
trace(result.digits); // 123
```

Es folgt ein weiteres Beispiel mit zwei benannten Gruppen mit den Bezeichnern `name` und `dom`:

```
var emailPattern:RegExp =
    /(?P<name>(\w|[_.\-])+)@(?P<dom>((\w|-)+))+\.\w{2,4}+/;
var address:String = "bob@example.com";
var result:Array = emailPattern.exec(address);
trace(result.name); // bob
trace(result.dom); // example
```

Hinweis: *Benannte Gruppen sind nicht Bestandteil der ECMAScript-Sprachspezifikation. Es handelt sich um eine Funktionserweiterung von ActionScript 3.0.*

Flags und Eigenschaften

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der folgenden Tabelle sind die fünf Flags aufgeführt, die für reguläre Ausdrücke gesetzt werden können. Jedes Flag kann als Eigenschaft des regulären Ausdrucks abgerufen werden.

Flag	Eigenschaft	Beschreibung
`g`	`global`	Entspricht mehreren Übereinstimmungen.
`i`	`ignoreCase`	Bei der Suche nach Übereinstimmungen wird die Groß- und Kleinschreibung nicht beachtet. Wird für die Zeichen `A-Z` und `a-z`, jedoch nicht für Sonderzeichen wie `Ä` und `ä` angewendet.
`m`	`multiline`	Wenn dieses Flag gesetzt ist, entsprechen `$` und `^` jeweils dem Anfang einer Zeile bzw. dem Ende einer Zeile.
`s`	`dotall`	Wenn dieses Flag gesetzt ist, werden mit `.` (Punkt) auch Zeilenvorschubzeichen (`\n`) gefunden.
`x`	`extended`	Ermöglicht erweiterte reguläre Ausdrücke. Sie können in einem regulären Ausdruck Leerzeichen eingeben, die nicht als Bestandteil des Musters aufgefasst werden. Dadurch kann der Code für den regulären Ausdruck besser lesbar eingegeben werden.

Beachten Sie, dass diese Eigenschaften schreibgeschützt sind. Sie können die Flags (`g`, `i`, `m`, `s` und `x`) wie folgt beim Definieren einer Variablen für einen regulären Ausdruck setzen:

```
var re:RegExp = /abc/gimsx;
```

Die benannten Eigenschaften können jedoch nicht direkt gesetzt werden. Der folgende Code führt beispielsweise zu einer Fehlermeldung:

```
var re:RegExp = /abc/;
re.global = true; // This generates an error.
```

In der Standardeinstellung sind die Flags nicht gesetzt, es sei denn, Sie geben sie in der Deklaration für einen regulären Ausdruck an. Die entsprechenden Eigenschaften sind ebenfalls auf `false` gesetzt.

Es sind darüber hinaus zwei weitere Eigenschaften für reguläre Ausdrücke verfügbar:

- Mit der `lastIndex`-Eigenschaft wird die Indexposition im String angegeben, die für den nächsten Aufruf der Methoden `exec()` oder `test()` eines regulären Ausdrucks verwendet wird.
- Mit der `source`-Eigenschaft wird der String angegeben, der das Muster in einem regulären Ausdruck definiert.

g-Flag (global)

Wenn das `g`-Flag (`global`) *nicht* gesetzt ist, wird je regulärem Ausdruck nur eine Übereinstimmung gefunden. Wenn das `g`-Flag im regulären Ausdruck nicht gesetzt ist, gibt die `String.match()`-Methode beispielsweise nur einen übereinstimmenden Teilstring zurück:

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/;
trace(str.match(pattern)) // output: she
```

Wenn das `g`-Flag gesetzt ist, gibt die `String.match()`-Methode wie folgt mehrere Übereinstimmungen zurück:

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/g;
// The same pattern, but this time the g flag IS set.
trace(str.match(pattern)); // output: she,shells,shore
```

i-Flag (ignoreCase)

In der Standardeinstellung wird bei Übereinstimmungen regulärer Ausdrücke die Groß- und Kleinschreibung beachtet. Wenn Sie das `i`-Flag (`ignoreCase`) setzen, wird die Groß- und Kleinschreibung ignoriert. Das kleingeschriebene `s` im regulären Ausdruck entspricht beispielsweise nicht dem ersten Zeichen des Strings, dem großgeschriebenen Buchstaben `S`:

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/)); // output: 13 -- Not the first character
```

Wenn das `i`-Flag jedoch gesetzt ist, stimmt der Großbuchstabe `S` mit dem regulären Ausdruck überein:

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/i)); // output: 0
```

Mit dem `i`-Flag wird die Groß- und Kleinschreibung nur bei den Zeichen `A-Z` und `a-z`, jedoch nicht bei Sonderzeichen wie `Ä` und `ä` ignoriert.

m-Flag (multiline)

Wenn das `m`-Flag (`multiline`) nicht gesetzt ist, entspricht `^` dem Anfang eines Strings und `$` dem Ende eines Strings. Wenn das `m`-Flag gesetzt ist, entsprechen diese Zeichen jeweils dem Anfang bzw. dem Ende einer Zeile im String. Betrachten Sie den folgenden String, der ein Zeilenvorschubzeichen enthält:

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/g)); // Match a word at the beginning of the string.
```

Obwohl das `g`-Flag (`global`) im regulären Ausdruck gesetzt ist, gibt die `match()`-Methode nur einen übereinstimmenden Teilstring zurück, da nur eine Entsprechung für das `^`-Zeichen vorliegt, nämlich am Anfang des Strings. Folgendes wird ausgegeben:

```
Test
```

Es folgt der gleiche Code mit dem gesetzten `m`-Flag:

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/gm)); // Match a word at the beginning of lines.
```

Nun werden die Wörter am Anfang der beiden Zeilen ausgegeben:

```
Test,Multiline
```

Beachten Sie, dass nur das Zeichen `\n` das Ende einer Zeile angibt. Das Ende einer Zeile wird jedoch nicht durch folgende Zeichen angegeben:

- Wagenrücklaufzeichen (`\r`)
- Unicode-Zeilentrennzeichen (`\u2028`)
- Unicode-Absatztrennzeichen (`\u2029`)

s-Flag (dotall)

Wenn das `s`-Flag (`dotall` oder „dot all“) nicht gesetzt ist, entspricht ein Punkt (`.`) in einem regulären Ausdruck nicht dem Zeilenvorschubzeichen (`\n`). Daher wird im folgenden Beispiel keine Entsprechung zurückgegeben:

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*?<\/p>/;
trace(str.match(re));
```

Wenn das `s`-Flag jedoch gesetzt ist, wird mit dem Punkt auch eine Übereinstimmung mit dem Zeilenvorschubzeichen erkannt:

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*?<\/p>/s;
trace(str.match(re));
```

In diesem Fall stimmt der gesamte Teilstring innerhalb der `<p>`-Tags mit dem regulären Ausdruck überein, einschließlich des Zeilenvorschubzeichens:

```
<p>Test
Multiline</p>
```

x-Flag (extended)

Reguläre Ausdrücke sind unter Umständen schwer lesbar, vor allem, wenn sie viele Metasymbole und Metasequenzen enthalten. Zum Beispiel:

```
/<p(>|(\s*[^>]*>)).*?<\/p>/gi
```

Wenn Sie das `x`-Flag (`extended`) in einem regulären Ausdruck verwenden, werden alle im Muster eingefügten Leerzeichen ignoriert. Der folgende reguläre Ausdruck ist beispielsweise mit dem vorherigen Beispiel identisch:

```
/     <p    (>  |  (\s*  [^>]*  >))    .*?    <\/p>  /gix
```

Wenn das `x`-Flag gesetzt ist und Sie angeben möchten, dass ein bestimmtes Leerzeichen nicht ignoriert werden soll, fügen Sie vor dem Leerzeichen einen umgekehrten Schrägstrich ein. Die beiden folgenden regulären Ausdrücke sind beispielsweise gleichwertig:

```
/foo bar/
/foo \ bar/x
```

lastIndex-Eigenschaft

Die `lastIndex`-Eigenschaft gibt die Indexposition im String an, ab der die nächste Suche beginnen soll. Diese Eigenschaft hat Auswirkungen auf die Methoden `exec()` und `test()`, die für einen regulären Ausdruck aufgerufen werden, bei dem das `g`-Flag auf `true` gesetzt ist. Betrachten Sie den folgenden Beispielcode:

```
var pattern:RegExp = /p\w*/gi;
var str:String = "Pedro Piper picked a peck of pickled peppers.";
trace(pattern.lastIndex);
var result:Object = pattern.exec(str);
while (result != null)
{
    trace(pattern.lastIndex);
    result = pattern.exec(str);
}
```

Die `lastIndex`-Eigenschaft ist in der Standardeinstellung auf 0 gesetzt (damit Suchvorgänge am Anfang eines Strings gestartet werden). Nach jeder Übereinstimmung wird die Eigenschaft auf die Indexposition nach der Übereinstimmung gesetzt. Dies ergibt die folgende Ausgabe für das vorherige Codebeispiel:

```
0
5
11
18
25
36
44
```

Wenn das `global`-Flag auf `false` gesetzt ist, verwenden oder setzen die Methoden `exec()` und `test()` die `lastIndex`-Eigenschaft nicht.

Die Methoden `match()`, `replace()` und `search()` der String-Klasse starten alle Suchvorgänge am Anfang eines Strings, unabhängig von der Einstellung der `lastIndex`-Eigenschaft des regulären Ausdrucks, der zum Aufrufen der entsprechenden Methode verwendet wird. (Mit der `match()`-Methode wird die `lastIndex`-Eigenschaft jedoch auf 0 gesetzt.)

Sie können die Eigenschaft `lastIndex` setzen, um die Anfangsposition in dem String zu setzen, in dem nach einem regulären Ausdruck gesucht werden soll.

source-Eigenschaft

Mit der `source`-Eigenschaft wird der String angegeben, der das Muster in einem regulären Ausdruck definiert. Zum Beispiel:

```
var pattern:RegExp = /foo/gi;
trace(pattern.source); // foo
```

Methoden für das Verwenden regulärer Ausdrücke bei Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die RegExp-Klasse umfasst zwei Methoden: `exec()` und `test()`.

Zusätzlich zu den Methoden `exec()` und `test()` der RegExp-Klasse enthält die String-Klasse die folgenden Methoden, mit denen Sie reguläre Ausdrücke in Strings suchen können: `match()`, `replace()`, `search()`, and `splice()`.

test()-Methode

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `test()`-Methode der RegExp-Klasse wird überprüft, ob der angegebene String eine Übereinstimmung mit dem regulären Ausdruck enthält, wie im folgenden Beispiel dargestellt:

```
var pattern:RegExp = /Class-\w/;
var str = "Class-A";
trace(pattern.test(str)); // output: true
```

exec()-Methode

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `exec()`-Methode der RegExp-Klasse überprüft den angegebenen String auf eine Übereinstimmung mit dem regulären Ausdruck und gibt ein Array mit den folgenden Elementen zurück:

- übereinstimmender Teilstring
- übereinstimmende Teilstrings aller in Klammern eingeschlossenen Gruppen im regulären Ausdruck

Das Array enthält zudem eine `index`-Eigenschaft, mit der die Indexposition am Anfang des übereinstimmenden Teilstrings angegeben wird.

Betrachten Sie den folgenden Beispielcode:

```
var pattern:RegExp = /\d{3}\-\d{3}-\d{4}/; //U.S phone number
var str:String = "phone: 415-555-1212";
var result:Array = pattern.exec(str);
trace(result.index, " - ", result);
// 7-415-555-1212
```

Führen Sie die `exec()`-Methode mehrfach aus, um mehrere Teilstrings zu suchen, wenn im regulären Ausdruck das `g`-Flag (`global`) gesetzt ist:

```
var pattern:RegExp = /\w*sh\w*/gi;
var str:String = "She sells seashells by the seashore";
var result:Array = pattern.exec(str);

while (result != null)
{
    trace(result.index, "\t", pattern.lastIndex, "\t", result);
    result = pattern.exec(str);
}
//output:
// 0  3  She
// 10  19  seashells
// 27  35  seashore
```

String-Methoden mit RegExp-Parametern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei den folgenden Methoden der String-Klasse können reguläre Ausdrücke als Parameter verwendet werden: `match()`, `replace()`, `search()` und `split()`. Weitere Informationen zu diesen Methoden finden Sie unter „Suchen von Mustern in Strings und Ersetzen von Teilstrings“ auf Seite 17.

Beispiel für reguläre Ausdrücke: Wiki-Parser

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit einem einfachen Beispiel für die Wiki-Textkonvertierung werden mehrere Verwendungsmöglichkeiten regulärer Ausdrücke veranschaulicht:

- Konvertieren von Textzeilen, die mit einem Wiki-Quellmuster übereinstimmen, in entsprechende HTML-Ausgabestrings
- Verwenden eines regulären Ausdrucks zum Konvertieren von URL-Mustern in `<a>`-Tags für HTML-Hyperlinks.
- Verwenden eines regulären Ausdrucks zum Konvertieren von US-Dollar-Strings (z. B. „`$9.95`") in Euro-Strings (z. B. „`8.24 €`").

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „WikiEditor“ befinden sich im Ordner „Samples/WikiEditor“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
WikiEditor.mxml oder WikiEditor.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/regExpExamples/WikiParser.as	Eine Klasse mit Methoden, mit denen Wiki-Eingabetextmuster über reguläre Ausdrücke in die entsprechende HTML-Ausgabe konvertiert werden.
com/example/programmingas3/regExpExamples/URLParser.as	Eine Klasse mit Methoden, mit denen URL-Strings über reguläre Ausdrücke in `<a>`-Tags für HTML-Hyperlinks konvertiert werden.
com/example/programmingas3/regExpExamples/CurrencyConverter.as	Eine Klasse mit Methoden, mit denen US-Dollar-Strings über reguläre Ausdrücke in Euro-Strings konvertiert werden.

Definieren der WikiParser-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die WikiParser-Klasse enthält Methoden zum Konvertieren von Wiki-Eingabetext in die entsprechende HTML-Ausgabe. Dies ist keine sehr stabile Anwendung für die Wiki-Konvertierung. Mit dieser Anwendung werden jedoch einige nützliche Einsatzmöglichkeiten von regulären Ausdrücken zum Suchen nach Übereinstimmungen mit Mustern und zum Konvertieren von Strings veranschaulicht.

Die Konstruktorfunktion in Verbindung mit der `setWikiData()`-Methode initialisiert wie folgt einen Beispielstring für den Wiki-Eingabetext:

```
public function WikiParser()
{
    wikiData = setWikiData();
}
```

Wenn der Benutzer in der Beispielanwendung auf die Schaltfläche „Test“ klickt, wird die `parseWikiString()`-Methode des WikiParser-Objekts aufgerufen. In dieser Methode werden mehrere andere Methoden aufgerufen, mit denen der resultierende HTML-String zusammengesetzt wird.

```
public function parseWikiString(wikiString:String):String
{
    var result:String = parseBold(wikiString);
    result = parseItalic(result);
    result = linesToParagraphs(result);
    result = parseBullets(result);
    return result;
}
```

Bei jeder der aufgerufenen Methoden – `parseBold()`, `parseItalic()`, `linesToParagraphs()` und `parseBullets()` – werden die durch einen regulären Ausdruck definierten übereinstimmenden Muster mit der `replace()`-Methode des Strings ersetzt, sodass der Wiki-Eingabetext in Text im HTML-Format umgewandelt wird.

Konvertieren von Mustern mit Fettdruck und Kursivdruck

Die `parseBold()`-Methode sucht Wiki-Textmuster mit Fettdruck (z. B. `'''foo'''`) und wandelt diese in das entsprechende HTML-Format um (z. B. `<b>foo</b>`), wie im Folgenden dargestellt:

```
private function parseBold(input:String):String
{
    var pattern:RegExp = /'''(.*?)'''/g;
    return input.replace(pattern, "<b>$1</b>");
}
```

Beachten Sie, dass der Teil `(.?*)` des regulären Ausdrucks einer beliebigen Anzahl Zeichen (`*`) zwischen den beiden begrenzenden `'''`-Mustern entspricht. Durch den `?`-Quantifizierer wird eine genügsame Suche durchgeführt, sodass bei einem String wie `'''aaa''' bbb '''ccc'''` der erste übereinstimmende String `'''aaa'''` ist und nicht der gesamte String (der mit einem `'''`-Muster beginnt und endet).

Durch die Klammern im regulären Ausdruck wird eine zwischengespeicherte Gruppe definiert. Die `replace()`-Methode verweist mit dem `$1`-Code im Ersetzungsstring auf diese Gruppe. Mit dem `g`-Flag (`global`) im regulären Ausdruck wird sichergestellt, dass mit der `replace()`-Methode alle Übereinstimmungen (und nicht nur die erste Übereinstimmung) im String ersetzt werden.

Die `parseItalic()`-Methode ähnelt der `parseBold()`-Methode, mit dem Unterschied, dass mit zwei Apostrophen (`''`) (und nicht mit drei Apostrophen) nach dem Trennzeichen für kursiven Text gesucht wird:

```
private function parseItalic(input:String):String
{
    var pattern:RegExp = /''(.*?)''/g;
    return input.replace(pattern, "<i>$1</i>");
}
```

Konvertieren von Mustern mit Aufzählungszeichen

Wie im folgenden Beispiel dargestellt ist, wird mit der `parseBullet()`-Methode nach dem Wiki-Muster mit Zeilen mit Aufzählungszeichen (z. B. `* foo`) gesucht und dieses Muster in die HTML-Entsprechung umgewandelt (z. B. `<li>foo</li>`):

```
private function parseBullets(input:String):String
{
    var pattern:RegExp = /^\*(.*)/gm;
    return input.replace(pattern, "<li>$1</li>");
}
```

Das `^`-Symbol am Anfang des regulären Ausdrucks entspricht dem Anfang einer Zeile. Durch das `m`-Flag (`multiline`) im regulären Ausdruck entspricht das `^`-Symbol dem Zeilenanfang und nicht dem Anfang des ganzen Strings.

Das Muster `\*` entspricht einem Sternchen. (Durch den umgekehrten Schrägstrich wird ein Sternchen und kein `*`-Quantifizierer angegeben.)

Durch die Klammern im regulären Ausdruck wird eine zwischengespeicherte Gruppe definiert. Die `replace()`-Methode verweist mit dem `$1`-Code im Ersetzungsstring auf diese Gruppe. Mit dem `g`-Flag (`global`) im regulären Ausdruck wird sichergestellt, dass mit der `replace()`-Methode alle Übereinstimmungen (und nicht nur die erste Übereinstimmung) im String ersetzt werden.

Konvertieren von Wiki-Absatzmustern

Mit der `linesToParagraphs()`-Methode wird jede Zeile im Wiki-Eingabestring in das HTML-Tag `<p>` für Absätze konvertiert. Mit diesen Zeilen in der Methode werden leere Zeilen aus dem Wiki-Eingabestring entfernt:

```
var pattern:RegExp = /^$/gm;
var result:String = input.replace(pattern, "");
```

Das `^`-Symbol und das `$`-Symbol im regulären Ausdruck entspricht dem Anfang bzw. dem Ende einer Zeile. Durch das `m`-Flag (`multiline`) im regulären Ausdruck entspricht das `^`-Symbol dem Zeilenanfang und nicht dem Anfang des ganzen Strings.

Mit der `replace()`-Methode werden alle übereinstimmenden Teilstrings (leere Zeilen) jeweils durch einen Leerstring (`""`) ersetzt. Mit dem `g`-Flag (`global`) im regulären Ausdruck wird sichergestellt, dass mit der `replace()`-Methode alle Übereinstimmungen (und nicht nur die erste Übereinstimmung) im String ersetzt werden.

Konvertieren von URLs in <a>-HTML-Tags

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn der Benutzer in der Beispielanwendung auf die Schaltfläche „Test“ klickt und zuvor das Kontrollkästchen „`urlToATag` aktiviert hat, wird die statische `URLParser.urlToATag()`-Methode aufgerufen, um URL-Strings aus dem Wiki-Eingabestring in `<a>`-HTML-Tags umzuwandeln.

```
var protocol:String = "((?:http|ftp)://)";
var urlPart:String = "([a-z0-9_-]+\.[a-z0-9_-]+)";
var optionalUrlPart:String = "(\.[a-z0-9_-]*)";
var urlPattern:RegExp = new RegExp(protocol + urlPart + optionalUrlPart, "ig");
var result:String = input.replace(urlPattern, "<a href='$1$2$3'><u>$1$2$3</u></a>");
```

Mithilfe der `RegExp()`-Konstruktorfunktion wird ein regulärer Ausdruck (`urlPattern`) aus mehreren Bestandteilen gebildet. Diese Bestandteile sind alle Strings, mit denen Teile des regulären Ausdrucks definiert werden.

Der erste Teil des regulären Ausdrucks, der durch den `protocol`-String definiert wird, legt ein URL-Protokoll fest: `http://` oder `ftp://`. Mit den Klammern wird eine nicht zwischengespeicherte Gruppe festgelegt, die durch das `?`-Symbol angegeben wird. Dies bedeutet, dass die Klammern nur zum Definieren einer Gruppe für das `|`-Auswahlmuster verwendet werden. Die Gruppe entspricht keinem der Rückverweiscodes (`$1`, `$2`, `$3`) im Ersetzungsstring der `replace()`-Methode.

Die anderen Bestandteile des regulären Ausdrucks verwenden jeweils zwischengespeicherte Gruppen (angegeben durch Klammern im Muster), die dann in den Rückverweiscodes (`$1`, `$2`, `$3`) im Ersetzungsstring der `replace()`-Methode verwendet werden.

Der Teil des Musters, der über den `urlPart`-String definiert wird, entspricht *mindestens* einem der folgenden Zeichen: `a-z`, `0-9`, _ oder `-`. Mit dem `+`-Quantifizierer wird angegeben, dass mindestens eines dieser Zeichen übereinstimmen muss. Mit `\.` wird ein erforderlicher Punkt (`.`) angegeben. Der restliche Teil entspricht einem anderen String mit mindestens einem der folgenden Zeichen: `a-z`, `0-9`, _ oder `-`.

Der Teil des Musters, der über den `optionalUrlPart`-String definiert wird, entspricht *keinem, einem oder mehreren* der folgenden Zeichen: einem Punkt (`.`), gefolgt von einer beliebigen Anzahl alphanumerischer Zeichen (einschließlich _ und `-`). Mit dem `*`-Quantifizierer wird angegeben, dass nach Übereinstimmungen mit keinem, einem oder mehreren dieser Zeichen gesucht wird.

Beim Aufrufen der `replace()`-Methode wird der reguläre Ausdruck verwendet und der HTML-Ersetzungsstring über Rückverweise zusammengesetzt.

Mit der `urlToATag()`-Methode wird dann die `emailToATag()`-Methode aufgerufen, mit der auf ähnliche Weise E-Mail-Muster durch `<a>`-Tags für Hyperlinkstrings ersetzt werden. Die regulären Ausdrücke für die Entsprechung mit HTTP-, FTP- und E-Mail-URLs in dieser Beispieldatei sind relativ einfach gehalten. Es sind sehr viel kompliziertere reguläre Ausdrücke für die korrekte Suche nach diesen URLs erforderlich.

Konvertieren von US-Dollar-Strings in Euro-Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn der Benutzer in der Beispielanwendung auf die Schaltfläche „Test“ klickt und zuvor das Kontrollkästchen `dollarToEuro` aktiviert hat, wird die statische `CurrencyConverter.usdToEuro()`-Methode aufgerufen, um US-Dollar-Strings (z. B. „`$9.95`") wie folgt in Euro-Strings (z. B. „`8.24 €`") umzuwandeln:

```
var usdPrice:RegExp = /\$([\d,]+.\d+)+/g;
return input.replace(usdPrice, usdStrToEuroStr);
```

In der ersten Zeile ist ein einfaches Muster für US-Dollar-Strings definiert. Beachten Sie, dass dem `$`-Zeichen ein umgekehrter Schrägstrich (`\`) als Escape-Zeichen vorangestellt ist.

Die `replace()`-Methode verwendet den regulären Ausdruck zum Suchen von Mustern und ruft die `usdStrToEuroStr()`-Funktion auf, um den Ersetzungsstring zu ermitteln (ein Wert in Euro).

Wenn als zweiter Parameter der `replace()`-Methode der Name einer Funktion verwendet wird, werden folgende Elemente als Parameter an die aufgerufene Funktion übergeben:

- Der übereinstimmende Teil des Strings
- Alle übereinstimmenden zwischengespeicherten (in Klammern eingeschlossenen) Gruppen. Die Anzahl der auf diese Weise übergebenen Argumente hängt von der Anzahl der Übereinstimmungen mit einer zwischengespeicherten Gruppe ab. Sie können die Anzahl der Übereinstimmungen mit zwischengespeicherten Gruppen ermitteln, indem Sie im Funktionscode `arguments.length - 3` überprüfen.
- Die Indexposition im String, an der die Übereinstimmung beginnt.
- Der vollständige String.

Mit der `usdStrToEuroStr()`-Methode werden US-Dollar-Strings wie folgt in Euro-Strings konvertiert:

```
private function usdToEuro(...args):String
{
    var usd:String = args[1];
    usd = usd.replace(",", "");
    var exchangeRate:Number = 0.828017;
    var euro:Number = Number(usd) * exchangeRate;
    trace(usd, Number(usd), euro);
    const euroSymbol:String = String.fromCharCode(8364); // €
    return euro.toFixed(2) + " " + euroSymbol;
}
```

Beachten Sie, dass `args[1]` die zwischengespeicherte Gruppe angibt, die dem regulären Ausdruck `usdPrice` entspricht. Dies ist der numerische Teil des US-Dollar-Strings, d. h. der Dollarbetrag ohne das `$`-Zeichen. Mit der Methode wird eine Währungsumrechnung durchgeführt und der resultierende String zurückgegeben (mit einem nachgestellten €-Symbol anstelle des vorangestellten $-Symbols).

Kapitel 6: XML-Verarbeitung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 enthält eine Gruppe von Klassen, die auf der E4X-Spezifikation (ECMAScript for XML, ECMA-357 Version 2) beruhen. Diese Klassen bieten leistungsstarke und benutzerfreundliche Funktionen für die Verarbeitung von XML-Daten. Mit E4X können Sie Code mit XML-Daten wesentlich schneller entwickeln, als dies mit den bisherigen Programmiertechniken möglich war. Ein zusätzlicher Vorteil besteht darin, dass der entstehende Code besser lesbar ist.

Verwandte Hilfethemen

XML-Klasse

ECMA-357-Spezifikation

Grundlagen von XML

Flash Player 9 und höher, Adobe AIR 1.0 und höher

XML ist eine Standardmethode zum Darstellen strukturierter Informationen, damit sie problemlos von Computern verarbeitet und relativ einfach von Benutzern geschrieben und verstanden werden können. XML ist die Abkürzung für eXtensible Markup Language (erweiterbare Auszeichnungssprache). Der XML-Standard ist unter www.w3.org/XML/ abrufbar.

XML bietet eine standardisierte und bequeme Methode zum Kategorisieren von Daten, um das Lesen, Abrufen und Bearbeiten dieser Daten zu erleichtern. Bei XML wird eine Baum- und Tagstruktur verwendet, die HTML ähnelt. Es folgt ein einfaches Beispiel für XML-Daten:

```
<song>
    <title>What you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

XML-Daten können auch komplexer sein, mit in anderen Tags verschachtelten Tags sowie Attributen und weiteren strukturellen Komponenten. Es folgt ein komplexeres Beispiel für XML-Daten:

```
<album>
    <title>Questions, unanswered</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <tracks>
        <song tracknumber="1" length="4:05">
            <title>What do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:31</lastplayed>
        </song>
        <song tracknumber="2" length="3:45">
            <title>Who do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:35</lastplayed>
        </song>
        <song tracknumber="3" length="5:14">
            <title>When do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:39</lastplayed>
        </song>
        <song tracknumber="4" length="4:19">
            <title>Do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:44</lastplayed>
        </song>
    </tracks>
</album>
```

Beachten Sie, dass dieses XML-Dokument intern weitere vollständige XML-Strukturen enthält (beispielsweise die `song`-Tags mit ihren untergeordneten Elementen). Es veranschaulicht auch weitere XML-Strukturen wie Attribute (`tracknumber` und `length` in den `song`-Tags) sowie Tags, die anstelle von Daten weitere Tags enthalten (z. B. das `tracks`-Tag).

Erste Schritte mit XML

Für den Fall, dass Sie bisher nur wenig oder gar keine Erfahrungen mit XML gesammelt haben, folgt hier eine kurze Beschreibung der gebräuchlichsten Aspekte von XML-Daten. XML-Daten sind Klartext mit einer bestimmten Syntax zum Ordnen der Informationen in einem strukturierten Format. Allgemein wird ein einzelner Satz XML-Daten als *XML-Dokument* bezeichnet. Im XML-Format werden Daten mithilfe einer hierarchischen Struktur in *Elemente* unterteilt (die aus einzelnen Datenelementen oder aus Containern für weitere Elemente bestehen können). Jedes XML-Dokument weist auf der obersten Ebene ein einzelnes Element auf – das Hauptelement. Innerhalb dieses Stammelements kann eine einzelne Informationsangabe abgelegt sein. In der Regel befinden sich dort jedoch weitere Elemente, die wiederum Elemente enthalten usw. Beispielsweise enthält das folgende XML-Dokument Informationen über ein Musikalbum:

```
<song tracknumber="1" length="4:05">
    <title>What do you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <mood>Happy</mood>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

Jedes Element ist durch zwei *Tags* gekennzeichnet. Sie enthalten den Namen des Elements in spitzen Klammern (Kleiner-als- und Größer-als-Zeichen). Das öffnende Tag, das den Beginn des Elements angibt, enthält den Namen des Elements:

```
<title>
```

Im schließenden Tag, das das Ende des Elements angibt, steht vor dem Namen des Elements ein Schrägstrich:

```
</title>
```

Wenn ein Element keinen Inhalt hat, kann es als leeres Element (auch als „selbst schließendes Element“ bezeichnet) dargestellt werden. In XML ist das folgende Element:

```
<lastplayed/>
```

identisch mit diesem Element:

```
<lastplayed></lastplayed>
```

Zusätzlich zum Inhalt des Elements zwischen dem öffnenden und dem schließenden Tag können Elemente auch weitere Werte (die sogenannten *Attribute*) enthalten, die im öffnenden Elementtag definiert werden. Beispielsweise definiert das folgende XML-Element ein einzelnes Attribut mit dem Namen `length` und dem Wert `4:19`:

```
<song length="4:19"></song>
```

Jedes XML-Element verfügt über einen Inhalt, der entweder aus einem einzelnen Wert, mindestens einem XML-Element oder einem leeren Wert (für ein leeres Element) besteht.

Weiterführende Informationen zu XML

Zur Vertiefung der Verwendung von XML liegt eine Reihe zusätzlicher Bücher und Ressourcen vor, einschließlich der folgenden Websites:

- W3Schools-XML-Tutorial: http://w3schools.com/xml/
- XMLpitstop-Tutorials, Diskussionsforen usw.: http://xmlpitstop.com/

ActionScript-Klassen für die Verwendung von XML

ActionScript 3.0 enthält mehrere Klassen, die zur Verarbeitung XML-strukturierter Informationen eingesetzt werden. Die beiden wichtigsten Klassen sind:

- XML: Gibt ein einzelnes XML-Element an, das ein XML-Dokument mit mehreren untergeordneten Elementen oder ein Einzelwertelement innerhalb eines Dokuments sein kann.
- XMLList: Gibt eine Gruppe von XML-Elementen an. Ein XMLList-Objekt wird für mehrere XML-Elemente verwendet, die sich in der Hierarchie des XML-Dokuments auf derselben Ebene befinden und Bestandteil desselben übergeordneten Elements sind. Eine XMLList-Instanz ist beispielsweise die einfachste Methode zum Verarbeiten der folgenden XML-Elemente (die sich in einem XML-Dokument befinden):

  ```
  <artist type="composer">Fred Wilson</artist>
  <artist type="conductor">James Schmidt</artist>
  <artist type="soloist">Susan Harriet Thurndon</artist>
  ```

Für anspruchsvollere Verwendungszwecke, bei denen XML-Namespaces verwendet werden, enthält ActionScript auch die Klassen „Namespace“ und „QName“. Weitere Informationen finden Sie unter „Verwenden von XML-Namespaces“ auf Seite 118.

Neben den integrierten Klassen für XML enthält ActionScript 3.0 zudem verschiedene Operatoren, die spezielle Funktionen für das Zugreifen auf und Bearbeiten von XML-Daten bereitstellen. Der Ansatz für die Verarbeitung von XML-Daten mithilfe dieser Klassen und Operatoren wird als E4X (ECMAScript for XML) bezeichnet und ist in der Spezifikation ECMA-357 Version 2 definiert.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die Ihnen beim Programmieren von XML-Verarbeitungsroutinen begegnen:

Element Ein einzelnes Objekt in einem XML-Dokument, das als Inhalt zwischen dem öffnenden und dem schließenden Tag (einschließlich der Tags selbst) definiert ist. XML-Elemente können Textdaten oder andere Elemente enthalten bzw. leer sein.

Leeres Element Ein XML-Element, das keine untergeordneten Elemente enthält. Leere Elemente werden häufig als selbst schließende Tags dargestellt (z. B. `<element/>`).

Dokument Eine einzelne XML-Struktur. Ein XML-Dokument kann eine beliebige Anzahl von Elementen (oder auch nur ein einziges leeres Element) enthalten. Es muss jedoch auf der obersten Ebene ein einziges Element aufweisen, das alle anderen Elemente des Dokuments enthält.

Knoten Eine andere Bezeichnung für ein XML-Element.

Attribut Ein benannter Wert, der einem Element zugeordnet ist und nicht als gesondertes, verschachteltes untergeordnetes Element, sondern im öffnenden Tag des Elements festgelegt wird (im Format `attributname="wert"`).

E4X-Ansatz der XML-Verarbeitung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die ECMAScript for XML-Spezifikation definiert eine Reihe von Klassen und Funktionen für die Verarbeitung von XML-Daten. Diese tragen in ihrer Gesamtheit die Bezeichnung *E4X*. ActionScript 3.0 enthält die folgenden E4X-Klassen: XML, XMLList, QName und Namespace.

Die Methoden, Eigenschaften und Operatoren der E4X-Klassen wurden mit der folgenden Zielstellung entwickelt:

- Einfachheit – Soweit möglich, erleichtert E4X das Programmieren und Verstehen von Code für die Verwendung von XML-Daten.
- Konsistenz – Die Methoden und die Logik hinter E4X sind sowohl in sich konsistent als auch konsistent mit anderen Teilen von ActionScript.
- Vertrautheit – Auf XML-Daten wird mit bekannten Operatoren wie dem Punktoperator (.) zugegriffen.

Hinweis: *In ActionScript 2.0 gibt es eine andere XML-Klasse. Diese wurde in ActionScript 3.0 in „XMLDocument“ umbenannt, damit sie nicht mit der XML-Klasse von ActionScript 3.0 kollidiert, die Bestandteil von E4X ist. Hauptsächlich zur Unterstützung älterer Anwendungen enthält ActionScript 3.0 im flash.xml-Paket die veralteten Klassen XMLDocument, XMLNode, XMLParser und XMLTag. Die neuen E4X-Klassen sind Kernklassen. Für ihre Verwendung muss kein Paket importiert werden. Einzelheiten zu den älteren ActionScript 2.0 XML-Klassen finden Sie im Eintrag zum flash.xml-Paket im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.*

Es folgt ein Beispiel zum Bearbeiten von Daten mit E4X:

```
var myXML:XML =
    <order>
        <item id='1'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>
```

Häufig werden in einer Anwendung XML-Daten aus einer externen Quelle geladen, z. B. aus einem Webservice oder einem RSS-Feed. Aus Gründen der Übersichtlichkeit weisen die hier aufgeführten Codebeispiele XML-Daten jedoch als Literale zu.

Wie im folgenden Codebeispiel dargestellt ist, enthält E4X einige intuitive Operatoren wie den Punktoperator (`.`) oder den Attributbezeichneroperator (`@`) für den Zugriff auf Eigenschaften und Attribute in XML:

```
trace(myXML.item[0].menuName); // Output: burger
trace(myXML.item.(@id==2).menuName); // Output: fries
trace(myXML.item.(menuName=="burger").price); // Output: 3.95
```

Mithilfe der `appendChild()`-Methode können Sie den XML-Daten einen neuen untergeordneten Knoten zuweisen, wie im folgenden Codeausschnitt dargestellt:

```
var newItem:XML =
    <item id="3">
        <menuName>medium cola</menuName>
        <price>1.25</price>
    </item>

myXML.appendChild(newItem);
```

Verwenden Sie die Operatoren `@` und `.` nicht nur zum Lesen von Daten, sondern auch zum Zuweisen von Daten, wie im Folgenden dargestellt:

```
myXML.item[0].menuName="regular burger";
myXML.item[1].menuName="small fries";
myXML.item[2].menuName="medium cola";

myXML.item.(menuName=="regular burger").@quantity = "2";
myXML.item.(menuName=="small fries").@quantity = "2";
myXML.item.(menuName=="medium cola").@quantity = "2";
```

Verwenden Sie wie folgt eine `for`-Schleife, um die XML-Knoten zu durchlaufen:

```
var total:Number = 0;
for each (var property:XML in myXML.item)
{
    var q:int = Number(property.@quantity);
    var p:Number = Number(property.price);
    var itemTotal:Number = q * p;
    total += itemTotal;
    trace(q + " " + property.menuName + " $" + itemTotal.toFixed(2))
}
trace("Total: $", total.toFixed(2));
```

XML-Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

XML-Objekte können XML-Elemente, Attribute, Kommentare, Verarbeitungsanweisungen oder Textelemente darstellen.

XML-Objekte werden als Objekte mit *einfachem Inhalt* oder mit *komplexem Inhalt* klassifiziert. XML-Objekte mit untergeordneten Knoten werden als Objekte mit komplexem Inhalt klassifiziert. Es wird von XML-Objekten mit einfachem Inhalt gesprochen, wenn es sich um eines der folgenden Objekte handelt: ein Attribut, einen Kommentar, eine Verarbeitungsanweisung oder einen Textknoten.

Bei dem folgenden XML-Objekt handelt es sich beispielsweise um ein XML-Objekt mit komplexem Inhalt, einschließlich eines Kommentars und einer Verarbeitungsanweisung:

```
XML.ignoreComments = false;
XML.ignoreProcessingInstructions = false;
var x1:XML =
    <order>
        <!--This is a comment. -->
        <?PROC_INSTR sample ?>
        <item id='1'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>
```

Wie im folgenden Beispiel dargestellt ist, können Sie nun mithilfe der Methoden `comments()` und `processingInstructions()` neue XML-Objekte, einen Kommentar und eine Verarbeitungsanweisung erstellen:

```
var x2:XML = x1.comments()[0];
var x3:XML = x1.processingInstructions()[0];
```

XML-Eigenschaften

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die XML-Klasse verfügt über fünf statische Eigenschaften:

- Mit den Eigenschaften `ignoreComments` und `ignoreProcessingInstructions` wird festgelegt, ob Kommentare und Verarbeitungsanweisungen beim Analysieren von XML-Objekten ignoriert werden.
- Mit der `ignoreWhitespace`-Eigenschaft wird festgelegt, ob Leerraumzeichen in Elementtags und eingebetteten Ausdrücken ignoriert werden, die nur durch Leerraumzeichen getrennt sind.
- Die Eigenschaften `prettyIndentu`nd `prettyPrinting` werden verwendet, um den Text zu formatieren, der von den Methoden `toString()` und `toXMLString()` der XML-Klasse zurückgegeben wird.

Einzelheiten zu diesen Eigenschaften finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

XML-Methoden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die folgenden Methoden ermöglichen die Bearbeitung der hierarchischen Struktur von XML-Objekten:

- `appendChild()`
- `child()`
- `childIndex()`
- `children()`
- `descendants()`
- `elements()`
- `insertChildAfter()`
- `insertChildBefore()`
- `parent()`
- `prependChild()`

Die folgenden Methoden ermöglichen die Verarbeitung von XML-Objektattributen:

- `attribute()`
- `attributes()`

Die folgenden Methoden ermöglichen die Verarbeitung von XML-Objekteigenschaften:

- `hasOwnProperty()`
- `propertyIsEnumerable()`
- `replace()`
- `setChildren()`

Die folgenden Methoden sind zum Verwenden qualifizierter Namen und Namespaces bestimmt:

- `addNamespace()`
- `inScopeNamespaces()`
- `localName()`
- `name()`
- `namespace()`
- `namespaceDeclarations()`
- `removeNamespace()`
- `setLocalName()`
- `setName()`
- `setNamespace()`

Die folgenden Methoden dienen zum Verarbeiten und Ermitteln bestimmter Typen von XML-Inhalten:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`

- `nodeKind()`
- `processingInstructions()`
- `text()`

Die folgenden Methoden dienen zum Konvertieren in Strings und zum Formatieren von XML-Objekten:

- `defaultSettings()`
- `setSettings()`
- `settings()`
- `normalize()`
- `toString()`
- `toXMLString()`

Es gibt einige zusätzliche Methoden:

- `contains()`
- `copy()`
- `valueOf()`
- `length()`

Einzelheiten zu diesen Methoden finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

XMLList-Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine XMLList-Instanz gibt eine willkürliche Zusammenstellung von XML-Objekten an. Sie kann ganze XML-Dokumente, XML-Fragmente oder das Ergebnis einer XML-Abfrage enthalten.

Die folgenden Methoden ermöglichen die Bearbeitung der hierarchischen Struktur von XMLList-Objekten:

- `child()`
- `children()`
- `descendants()`
- `elements()`
- `parent()`

Die folgenden Methoden ermöglichen die Verarbeitung von XMLList-Objektattributen:

- `attribute()`
- `attributes()`

Die folgenden Methoden ermöglichen die Verarbeitung von XMLList-Objekteigenschaften:

- `hasOwnProperty()`
- `propertyIsEnumerable()`

Die folgenden Methoden dienen zum Verarbeiten und Ermitteln bestimmter Typen von XML-Inhalten:

- `comments()`

- `hasComplexContent()`
- `hasSimpleContent()`
- `processingInstructions()`
- `text()`

Die folgenden Methoden dienen zum Konvertieren in Strings und zum Formatieren von XMLList-Objekten:

- `normalize()`
- `toString()`
- `toXMLString()`

Es gibt einige zusätzliche Methoden:

- `contains()`
- `copy()`
- `length()`
- `valueOf()`

Einzelheiten zu diesen Methoden finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Bei einem XMLList-Objekt mit genau einem XML-Element können Sie alle Eigenschaften und Methoden der XML-Klasse verwenden, da ein XMLList-Objekt mit einem XML-Element wie ein XML-Objekt behandelt wird. Beispielsweise können Sie im folgenden Code die `appendChild()`-Methode der XML-Klasse verwenden, da `doc.div` ein XMLList-Objekt mit nur einem Element ist:

```
var doc:XML =
        <body>
            <div>
                <p>Hello</p>
            </div>
        </body>;
doc.div.appendChild(<p>World</p>);
```

Eine Liste der XML-Eigenschaften und -Methoden finden Sie unter „XML-Objekte“ auf Seite 107.

Initialisieren von XML-Variablen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können einem XML-Objekt wie folgt ein XML-Literal zuweisen:

```
var myXML:XML =
    <order>
        <item id='1'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>
```

Wie im folgenden Codeauszug dargestellt ist, können Sie auch den `new`-Konstruktor verwenden, um aus einem String mit XML-Daten eine Instanz eines XML-Objekts zu erstellen:

```
var str:String = "<order><item id='1'><menuName>burger</menuName>"
                           + "<price>3.95</price></item></order>";
var myXML:XML = new XML(str);
```

Wenn die XML-Daten im String nicht korrekt strukturiert sind (z. B. wenn ein schließendes Tag fehlt), wird ein Laufzeitfehler angezeigt.

Sie können auch Daten als Verweis (aus anderen Variablen) in ein XML-Objekt übergeben, wie im folgenden Beispiel dargestellt ist:

```
var tagname:String = "item";
var attributename:String = "id";
var attributevalue:String = "5";
var content:String = "Chicken";
var x:XML = <{tagname} {attributename}={attributevalue}>{content}</{tagname}>;
trace(x.toXMLString())
    // Output: <item id="5">Chicken</item>
```

Verwenden Sie die URLLoader-Klasse, um XML-Daten von einer URL zu lesen, wie im folgenden Beispiel dargestellt ist:

```
import flash.events.Event;
import flash.net.URLLoader;
import flash.net.URLRequest;

var externalXML:XML;
var loader:URLLoader = new URLLoader();
var request:URLRequest = new URLRequest("xmlFile.xml");
loader.load(request);
loader.addEventListener(Event.COMPLETE, onComplete);

function onComplete(event:Event):void
{
    var loader:URLLoader = event.target as URLLoader;
    if (loader != null)
    {
        externalXML = new XML(loader.data);
        trace(externalXML.toXMLString());
    }
    else
    {
        trace("loader is not a URLLoader!");
    }
}
```

Verwenden Sie die XMLSocket-Klasse, um XML-Daten von einer Socketverbindung zu lesen. Weitere Informationen finden Sie im Eintrag zur XMLSocket-Klasse im Handbuch ActionScript 3.0 Reference for the Adobe Flash Platform.

Zusammenstellen und Transformieren von XML-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der `prependChild()`-Methode oder der `appendChild()`-Methode können Sie eine Eigenschaft am Anfang oder am Ende der Eigenschaftenliste eines XML-Objekts einfügen, wie im folgenden Beispiel dargestellt ist:

```
var x1:XML = <p>Line 1</p>
var x2:XML = <p>Line 2</p>
var x:XML = <body></body>
x = x.appendChild(x1);
x = x.appendChild(x2);
x = x.prependChild(<p>Line 0</p>);
    // x == <body><p>Line 0</p><p>Line 1</p><p>Line 2</p></body>
```

Verwenden Sie wie folgt die `insertChildBefore()`-Methode oder die `insertChildAfter()`-Methode, um eine Eigenschaft vor oder nach einer bestimmten anderen Eigenschaft einzufügen:

```
var x:XML =
    <body>
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
    </body>
var newNode:XML = <p>Paragraph 1.5</p>
x = x.insertChildAfter(x.p[0], newNode)
x = x.insertChildBefore(x.p[2], <p>Paragraph 1.75</p>)
```

Wie im folgenden Beispiel veranschaulicht wird, können Sie auch die geschweiften Klammern (`{` und `}`) als Operator verwenden, um beim Erstellen von XML-Objekten Daten als Verweis (aus anderen Variablen) zu übergeben:

```
var ids:Array = [121, 122, 123];
var names:Array = [["Murphy","Pat"], ["Thibaut","Jean"], ["Smith","Vijay"]]
var x:XML = new XML("<employeeList></employeeList>");

for (var i:int = 0; i < 3; i++)
{
    var newnode:XML = new XML();
    newnode =
        <employee id={ids[i]}>
            <last>{names[i][0]}</last>
            <first>{names[i][1]}</first>
        </employee>;

    x = x.appendChild(newnode)
}
```

Mithilfe des `=`-Operators können Sie einem XML-Objekt Eigenschaften und Attribute zuweisen:

```
var x:XML =
    <employee>
        <lastname>Smith</lastname>
    </employee>
x.firstname = "Jean";
x.@id = "239";
```

Auf diese Weise wird dem XML-Objekt `x` Folgendes zugewiesen:

```
<employee id="239">
    <lastname>Smith</lastname>
    <firstname>Jean</firstname>
</employee>
```

Mit den Operatoren „+“ und „+=“ lassen sich XMLList-Objekte verketten:

```
var x1:XML = <a>test1</a>
var x2:XML = <b>test2</b>
var xList:XMLList = x1 + x2;
xList += <c>test3</c>
```

Auf diese Weise wird dem XMLList-Objekt `xList` Folgendes zugewiesen:

```
<a>test1</a>
<b>test2</b>
<c>test3</c>
```

Durchsuchen von XML-Strukturen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eines der wichtigen Merkmale von XML ist die Fähigkeit, komplexe, verschachtelte Daten als linearen String von Textdaten bereitzustellen. Beim Laden von Daten in ein XML-Objekt werden diese in ActionScript analysiert und in ihrer hierarchischen Struktur im Speicher abgelegt (bei nicht korrekt strukturierten XML-Daten wird ein Laufzeitfehler ausgelöst).

Mit den Operatoren und Methoden der XML- und XMLList-Objekte ist es einfach, die Struktur von XML-Daten zu durchlaufen.

Verwenden Sie für den Zugriff auf die untergeordneten Eigenschaften von XML-Objekten den Punktoperator (.) und den Nachfolgerzugriffsoperator (..). Gegeben ist das folgende XML-Objekt:

```
var myXML:XML =
    <order>
        <book ISBN="0942407296">
            <title>Baking Extravagant Pastries with Kumquats</title>
            <author>
                <lastName>Contino</lastName>
                <firstName>Chuck</firstName>
            </author>
            <pageCount>238</pageCount>
        </book>
        <book ISBN="0865436401">
            <title>Emu Care and Breeding</title>
            <editor>
                <lastName>Case</lastName>
                <firstName>Justin</firstName>
            </editor>
            <pageCount>115</pageCount>
        </book>
    </order>
```

Das Objekt `myXML.book` ist ein XMLList-Objekt, das untergeordnete Eigenschaften des `myXML`-Objekts mit dem Namen `book` enthält. Diese beiden XML-Objekte stimmen mit den beiden `book`-Eigenschaften des `myXML`-Objekts überein.

Das Objekt `myXML..lastName` ist ein XMLList-Objekt, das alle Nachfolgereigenschaften mit dem Namen `lastName` enthält. Diese beiden XML-Objekte stimmen mit den beiden `lastName`-Eigenschaften des `myXML`-Objekts überein.

Das Objekt `myXML.book.editor.lastName` ist ein XMLList-Objekt, das alle untergeordneten Knoten mit dem Namen `lastName` von untergeordneten Knoten mit dem Namen `editor` von untergeordneten Knoten mit dem Namen `book` des `myXML`-Objekts enthält: in diesem Fall ein XMLList-Objekt mit nur einem XML-Objekt (die `lastName`-Eigenschaft mit dem Wert `Case`).

Zugreifen auf über- und untergeordnete Knoten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `parent()`-Methode gibt den übergeordneten Knoten eines XML-Objekts zurück.

Sie können die ordinalen Indexwerte einer untergeordneten Liste verwenden, um auf bestimmte untergeordnete Objekte zuzugreifen. Stellen Sie sich beispielsweise das XML-Objekt `myXML` vor, das über zwei untergeordnete Eigenschaften mit dem Namen `book` verfügt. Jeder der untergeordneten Eigenschaften mit dem Namen `book` ist eine Indexnummer zugeordnet:

```
myXML.book[0]
myXML.book[1]
```

Um auf bestimmte über zwei Stufen hinweg untergeordnete Eigenschaften zuzugreifen, können Sie Indexnummern für die jeweils untergeordneten Namen angeben:

```
myXML.book[0].title[0]
```

Wenn es jedoch nur eine untergeordnete Eigenschaft von `x.book[0]` mit dem Namen `title` gibt, können Sie den Indexverweis wie folgt weglassen:

```
myXML.book[0].title
```

Analog hierzu können Sie beide Indexverweise weglassen, wenn nur eine dem Objekt `x` untergeordnete Eigenschaft mit dem Namen „book“ vorliegt:

```
myXML.book.title
```

Wie im folgenden Beispiel dargestellt ist, können Sie die `child()`-Methode verwenden, um zu untergeordneten Eigenschaften mit Namen zu navigieren, die auf einer Variablen oder einem Ausdruck basieren:

```
var myXML:XML =
        <order>
            <book>
                <title>Dictionary</title>
            </book>
        </order>;

var childName:String = "book";

trace(myXML.child(childName).title) // output: Dictionary
```

Zugreifen auf Attribute

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Verwenden Sie das `@`-Symbol (den Attributbezeichneroperator) wie im folgenden Code dargestellt, um auf Attribute eines XML- oder XMLList-Objekts zuzugreifen:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@id); // 6401
```

Wie im folgenden Code dargestellt ist, können Sie das Platzhaltersymbol `*` zusammen mit dem `@`-Symbol verwenden, um auf alle Attribute eines XML- oder XMLList-Objekts zuzugreifen:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@*.toXMLString());
// 6401
// 233
```

Wie im folgenden Code dargestellt ist, können Sie mithilfe der `attribute()`-Methode oder der `attributes()`-Methode auf ein bestimmtes Attribut oder auf alle Attribute eines XML- oder XMLList-Objekts zugreifen:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.attribute("id")); // 6401
trace(employee.attribute("*").toXMLString());
// 6401
// 233
trace(employee.attributes().toXMLString());
// 6401
// 233
```

Beachten Sie, dass Sie für den Zugriff auf Attribute auch die im folgenden Beispiel dargestellte Syntax verwenden können:

```
employee.attribute("id")
employee["@id"]
employee.@["id"]
```

Jeder dieser Ausdrücke entspricht `employee.@id`. Die Syntax `employee.@id` ist jedoch der bevorzugte Ansatz.

Filtern nach Attribut- oder Elementwerten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die Klammernoperatoren `(` und `)` verwenden, um Elemente mit einem bestimmten Elementnamen oder Attributwert zu filtern. Gegeben ist das folgende XML-Objekt:

```
var x:XML =
    <employeeList>
        <employee id="347">
            <lastName>Zmed</lastName>
            <firstName>Sue</firstName>
            <position>Data analyst</position>
        </employee>
        <employee id="348">
            <lastName>McGee</lastName>
            <firstName>Chuck</firstName>
            <position>Jr. data analyst</position>
        </employee>
    </employeeList>
```

Die folgenden Ausdrücke sind alle gültig:

- `x.employee.(lastName == "McGee")` – Dies ist der zweite `employee`-Knoten.
- `x.employee.(lastName == "McGee").firstName` – Dies ist die `firstName`-Eigenschaft des zweiten `employee`-Knotens.
- `x.employee.(lastName == "McGee").@id` – Dies ist der Wert des `id`-Attributs des zweiten `employee`-Knotens.
- `x.employee.(@id == 347)` – Der erste `employee`-Knoten.
- `x.employee.(@id == 347).lastName` – Dies ist die `lastName`-Eigenschaft des ersten `employee`-Knotens.
- `x.employee.(@id > 300)` – Dies ist ein XMLList-Objekt mit beiden `employee`-Eigenschaften.
- `x.employee.(position.toString().search("analyst") > -1)` – Dies ist ein XMLList-Objekt mit beiden `position`-Eigenschaften.

Wenn Sie versuchen, anhand von Attributen oder Elementen zu filtern, die nicht vorhanden sind, wird eine Ausnahme ausgegeben. Die letzte Zeile des folgenden Codes erzeugt beispielsweise einen Fehler, da im zweiten `p`-Element kein `id`-Attribut vorhanden ist:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(@id == '123'));
```

Analog erzeugt die letzte Zeile des folgenden Codes einen Fehler, da keine `b`-Eigenschaft des zweiten `p`-Elements vorhanden ist:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(b == 'Bob'));
```

Um solche Fehler zu vermeiden, können Sie Eigenschaften mit übereinstimmenden Attributen und Elementen angeben, indem Sie wie im folgenden Code die Methoden `attribute()` und `elements()` verwenden:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(attribute('id') == '123'));
trace(doc.p.(elements('b') == 'Bob'));
```

Sie können auch wie im folgenden Code die `hasOwnProperty()`-Methode verwenden:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(hasOwnProperty('@id') && @id == '123'));
trace(doc.p.(hasOwnProperty('b') && b == 'Bob'));
```

Verwenden der for..in- und der for each..in-Anweisung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 enthält die `for..in`-Anweisung und die `for each..in`-Anweisung zum schrittweisen Durchlaufen von XMLList-Objekten. Gegeben sind beispielsweise das folgende XML-Objekt `myXML` und das XMLList-Objekt `myXML.item`. Das XMLList-Objekt `myXML.item` enthält die beiden `item`-Knoten des XML-Objekts.

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2' quantity='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>;
```

Die `for..in`-Anweisung ermöglicht es Ihnen, eine Gruppe von Eigenschaftsnamen eines XMLList-Objekts zu durchlaufen:

```
var total:Number = 0;
for (var pname:String in myXML.item)
{
    total += myXML.item.@quantity[pname] * myXML.item.price[pname];
}
```

Die `for each..in`-Anweisung ermöglicht es Ihnen, die Eigenschaften des XMLList-Objekts zu durchlaufen:

```
var total2:Number = 0;
for each (var prop:XML in myXML.item)
{
    total2 += prop.@quantity * prop.price;
}
```

Verwenden von XML-Namespaces

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Namespaces in einem XML-Objekt (oder XML-Dokument) bezeichnen den Typ der Daten, die das Objekt enthält. Beispielsweise deklarieren Sie beim Senden von XML-Daten an einen Webserver, auf dem das SOAP-Nachrichtenprotokoll verwendet wird, den Namespace im öffnenden XML-Tag:

```
var message:XML =
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <soap:Body xmlns:w="http://www.test.com/weather/">
            <w:getWeatherResponse>
                <w:tempurature >78</w:tempurature>
            </w:getWeatherResponse>
        </soap:Body>
    </soap:Envelope>;
```

Der Namespace verfügt über ein Präfix (`soap`) und einen URI, der den Namespace definiert (`http://schemas.xmlsoap.org/soap/envelope/`).

ActionScript 3.0 enthält die Namespace-Klasse für die Bearbeitung von XML-Namespaces. Bei dem XML-Objekt im vorangegangenen Beispiel können Sie die Namespace-Klasse wie folgt einsetzen:

```
var soapNS:Namespace = message.namespace("soap");
trace(soapNS); // Output: http://schemas.xmlsoap.org/soap/envelope/

var wNS:Namespace = new Namespace("w", "http://www.test.com/weather/");
message.addNamespace(wNS);
var encodingStyle:XMLList = message.@soapNS::encodingStyle;
var body:XMLList = message.soapNS::Body;

message.soapNS::Body.wNS::GetWeatherResponse.wNS::tempurature = "78";
```

Die XML-Klasse enthält die folgenden Methoden für die Verwendung mit Namespaces: `addNamespace()`, `inScopeNamespaces()`, `localName()`, `name()`, `namespace()`, `namespaceDeclarations()`, `removeNamespace()`, `setLocalName()`, `setName()` und `setNamespace()`.

Mit der `default xml namespace`-Direktive können Sie einen Standardnamespace für XML-Objekte zuweisen. Im folgenden Beispiel haben `x1` und `x2` denselben Standardnamespace:

```
var ns1:Namespace = new Namespace("http://www.example.com/namespaces/");
default xml namespace = ns1;
var x1:XML = <test1 />;
var x2:XML = <test2 />;
```

XML-Typumwandlung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können XML-Objekte und XMLList-Objekte in String-Werte konvertieren. Ebenso können Sie Strings in XML-Objekte und XMLList-Objekte konvertieren. Beachten Sie auch, dass alle XML-Attributwerte, XML-Namen und XML-Textwerte Strings sind. In den folgenden Abschnitten werden alle diese Arten der XML-Typkonvertierung behandelt.

Konvertieren von XML- und XMLList-Objekten in Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die XML- und XMLList-Klassen enthalten eine `toString()`-Methode und eine `toXMLString()`-Methode. Die `toXMLString()`-Methode gibt einen String zurück, der alle Tags, Attribute, Namespace-Deklarationen und Inhaltsdaten des XML-Objekts enthält. Bei XML-Objekten mit komplexem Inhalt (d. h. mit untergeordneten Elementen) bewirkt die `toString()`-Methode genau dasselbe wie die `toXMLString()`-Methode. Bei XML-Objekten mit einfachem Inhalt (mit nur einem Textelement) gibt die `toString()`-Methode nur den Textinhalt des Elements zurück, wie im folgenden Beispiel dargestellt ist:

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName.toXMLString());
    // <menuName>burger</menuName>
trace(myXML.item[0].menuName.toString());
    // burger
```

Wenn Sie die `trace()`-Methode verwenden, ohne `toString()` oder `toXMLString()` anzugeben, werden die Daten standardmäßig mithilfe der `toString()`-Methode konvertiert, wie im folgenden Codebeispiel dargestellt ist:

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName);
    // burger
```

Bei Verwendung der `trace()`-Methode zum Debuggen empfiehlt es sich jedoch in der Regel, die `toXMLString()`-Methode zu verwenden, damit die `trace()`-Methode umfassendere Daten ausgibt.

Konvertieren von Strings in XML-Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können wie folgt mit dem `new XML()`-Konstruktor ein XML-Objekt aus einem String erstellen:

```
var x:XML = new XML("<a>test</a>");
```

Beim Versuch, einen String in XML zu konvertieren, der ungültige und nicht korrekt strukturierte XML-Daten enthält, wird ein Laufzeitfehler ausgelöst:

```
var x:XML = new XML("<a>test"); // throws an error
```

Konvertieren von Attributwerten, Namen und Textwerten aus Strings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Alle XML-Attributwerte, XML-Namen und XML-Textwerte sind vom Datentyp String und müssen gegebenenfalls in andere Datentypen umgewandelt werden. Im folgenden Codebeispiel werden Textwerte beispielsweise mithilfe der `Number()`-Funktion in Zahlen umgewandelt:

```
var myXML:XML =
                        <order>
                            <item>
                                <price>3.95</price>
                            </item>
                            <item>
                                <price>1.00</price>
                            </item>
                        </order>;

var total:XML = <total>0</total>;
myXML.appendChild(total);

for each (var item:XML in myXML.item)
{
    myXML.total.children()[0] = Number(myXML.total.children()[0])
                                                + Number(item.price.children()[0]);
}
trace(myXML.total); // 4.95;
```

Wenn in diesem Code nicht die `Number()`-Funktion verwendet wird, wird der „+“-Operator als Stringverkettungsoperator interpretiert, und die Ausgabe der `trace()`-Methode in der letzten Zeile lautet wie folgt:

```
01.003.95
```

Lesen externer XML-Dokumente

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der URLLoader-Klasse können Sie XML-Daten von einer URL laden. Um den folgenden Code in einer Anwendung verwenden zu können, ersetzen Sie den Wert `XML_URL` im Beispiel durch eine gültige URL:

```
import flash.events.Event;
import flash.net.URLLoader;

var myXML:XML = new XML();
var XML_URL:String = "http://www.example.com/Sample3.xml";
var myXMLURL:URLRequest = new URLRequest(XML_URL);
var myLoader:URLLoader = new URLLoader(myXMLURL);
myLoader.addEventListener(Event.COMPLETE, xmlLoaded);

function xmlLoaded(event:Event):void
{
    myXML = XML(myLoader.data);
    trace("Data loaded.");
}
```

Sie können auch die XMLSocket-Klasse einsetzen, um eine asynchrone XML-Socketverbindung mit einem Server einzurichten. Weitere Informationen finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Beispiel für XML in ActionScript: Laden von RSS-Daten aus dem Internet

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Beispielanwendung „RSSViewer“ werden mehrere Funktionen für die Verwendung von XML in ActionScript erläutert, einschließlich folgender Funktionen:

- Verwenden von XML-Methoden zum Durchlaufen von XML-Daten im RSS-Format
- Verwenden von XML-Methoden zum Zusammenstellen von XML-Daten im HTML-Format zur Verwendung in einem Textfeld

Das RSS-Format zum Bereitstellen von Nachrichten per XML ist sehr verbreitet. Es folgt ein Beispiel für eine einfache RSS-Datendatei:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
    <title>Alaska - Weather</title>
    <link>http://www.nws.noaa.gov/alerts/ak.html</link>
    <description>Alaska - Watches, Warnings and Advisories</description>

    <item>
        <title>
            Short Term Forecast - Taiya Inlet, Klondike Highway (Alaska)
        </title>
        <link>
            http://www.nws.noaa.gov/alerts/ak.html#A18.AJKNK.1900
        </link>
        <description>
            Short Term Forecast Issued At: 2005-04-11T19:00:00
            Expired At: 2005-04-12T01:00:00 Issuing Weather Forecast Office
            Homepage: http://pajk.arh.noaa.gov
        </description>
</item>
    <item>
        <title>
            Short Term Forecast - Haines Borough (Alaska)
        </title>
            <link>
            http://www.nws.noaa.gov/alerts/ak.html#AKZ019.AJKNOWAJK.190000
        </link>
        <description>
            Short Term Forecast Issued At: 2005-04-11T19:00:00
            Expired At: 2005-04-12T01:00:00 Issuing Weather Forecast Office
            Homepage: http://pajk.arh.noaa.gov
        </description>
    </item>
</channel>
</rss>
```

Die Anwendung „SimpleRSS“ liest RSS-Daten aus dem Internet, strukturiert die Daten nach Schlagzeilen (Überschriften), Links und Beschreibungstexten und gibt diese Daten dann zurück. Mit der SimpleRSSUI-Klasse werden die Benutzeroberfläche bereitgestellt und die SimpleRSS-Klasse aufgerufen, in der die gesamte XML-Verarbeitung erfolgt.

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „RSSViewer“ befinden sich im Ordner „Samples/RSSViewer“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
RSSViewer.mxml oder RSSViewer.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/rssViewer/RSSParser.as	Eine Klasse mit Methoden, mit denen E4X zum Durchlaufen der RSS-Daten (im XML-Format) verwendet und eine entsprechende HTML-Darstellung erzeugt wird.
RSSData/ak.rss	Eine RSS-Beispieldatei. Die Anwendung ist so eingerichtet, dass RSS-Daten aus dem Internet mit einem durch Adobe gehosteten Flex-RSS-Eingabestrom gelesen werden. Sie können jedoch die Anwendung problemlos so ändern, dass RSS-Daten aus dem aufgeführten Dokument gelesen werden, in dem ein geringfügig anderes Schema als im Flex-RSS-Datenstrom verwendet wird.

Lesen und Strukturieren von XML-Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die RSSParser-Klasse enthält die `xmlLoaded()`-Methode, mit der die in der `rssXML`-Variablen gespeicherten RSS-Eingabedaten in einen String mit HTML-formatierten Ausgabedaten (`rssOutput`) konvertiert werden.

Am Anfang der Methode wird im Code der XML-Standardnamespace festgelegt, wenn die RSS-Quelldaten einen Standardnamespace enthalten:

```
if (rssXML.namespace("") != undefined)
{
    default xml namespace = rssXML.namespace("");
}
```

In den nächsten Zeilen wird der Inhalt der XML-Quelldaten in einer Schleife durchlaufen. Zudem werden alle Nachfolgereigenschaften mit dem Namen `item` überprüft.

```
for each (var item:XML in rssXML..item)
{
    var itemTitle:String = item.title.toString();
    var itemDescription:String = item.description.toString();
    var itemLink:String = item.link.toString();
    outXML += buildItemHTML(itemTitle,
                            itemDescription,
                            itemLink);
}
```

In den ersten drei Zeilen werden einfach nur Stringvariablen für die Titel-, Beschreibungs- und Linkeigenschaften der `item`-Eigenschaft in den XML-Daten festgelegt. In der nächsten Zeile wird dann die `buildItemHTML()`-Methode aufgerufen, um HTML-Daten in Form eines XMLList-Objekts abzurufen. Dabei dienen die drei neuen Stringvariablen als Parameter.

Zusammenstellen von XMLList-Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die HTML-Daten (ein XMLList-Objekt) haben das folgende Format:

```
<b>itemTitle</b>
<p>
    itemDescription
    <br />
    <a href="link">
        <font color="#008000">More...</font>
    </a>
</p>
```

In den ersten Zeilen der Methode wird der XML-Standardnamespace gelöscht:

```
default xml namespace = new Namespace();
```

Der Gültigkeitsbereich der `default xml namespace`-Direktive ist auf Funktionsblöcke beschränkt. Dies bedeutet, dass der Gültigkeitsbereich dieser Deklaration die `buildItemHTML()`-Methode ist.

In den folgenden Zeilen wird das XMLList-Objekt mithilfe der an die Funktion übergebenen Stringargumente zusammengestellt:

```
var body:XMLList = new XMLList();
body += new XML("<b>" + itemTitle + "</b>");
var p:XML = new XML("<p>" + itemDescription + "</p>");

var link:XML = <a></a>;
link.@href = itemLink; // <link href="itemLinkString"></link>
link.font.@color = "#008000";
        // <font color="#008000"></font></a>
        // 0x008000 = green
link.font = "More...";

p.appendChild(<br/>);
p.appendChild(link);
body += p;
```

Dieses XMLList-Objekt enthält Stringdaten, die für ein ActionScript-HTML-Textfeld geeignet sind.

In der `xmlLoaded()`-Methode wird der Rückgabewert der `buildItemHTML()`-Methode in einen String konvertiert:

```
XML.prettyPrinting = false;
rssOutput = outXML.toXMLString();
```

Extrahieren des Titels des RSS-Feeds und Senden eines benutzerdefinierten Ereignisses

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der `xmlLoaded()`-Methode wird anhand der Informationen in den RSS-XML-Quelldaten eine Stringvariable `rssTitle` festgelegt:

```
rssTitle = rssXML.channel.title.toString();
```

Die `xmlLoaded()`-Methode erzeugt schließlich ein Ereignis, durch das die Anwendung darüber benachrichtigt wird, dass die Daten analysiert wurden und nun verfügbar sind:

```
            dataWritten = new Event("dataWritten", true);
```

Kapitel 7: Verwenden der nativen JSON-Funktionalität

ActionScript 3.0 stellt eine native API zum Kodieren und Dekodieren von ActionScript-Objekten bereit. Dazu wird das Format JavaScript Object Notation (JSON) verwendet. Die JSON-Klasse und ergänzende Mitgliederfunktionen folgen den Spezifikationen der ECMA-262, 5. Edition, mit einigen Abweichungen.

Community-Mitglied Todd Anderson stellt einen Vergleich der nativen JSON-API und der JSON-Klasse „as3corelib“ eines Drittanbieters bereit. Siehe Working with Native JSON in Flash Player 11.

Verwandte Hilfethemen

JSON

Überblick über die JSON-API

Die ActionScript-JSON-API besteht aus der JSON-Klasse und `toJSON()`-Mitgliedfunktionen in einigen nativen Klassen. Bei Anwendungen, die eine benutzerdefinierte JSON-Kodierung für eine beliebige Klasse erfordern, stellt das ActionScript-Framework Möglichkeiten zum Überschreiben der Standardkodierung bereit.

Die JSON-Klasse verarbeitet intern den Import und Export für alle ActionScript-Klassen, die kein `toJSON()`-Mitglied bereitstellen. Für derartige Klassen arbeitet sich JSON durch die öffentlichen Eigenschaften aller Objekte, die erkannt werden. Wenn ein Objekt andere Objekte enthält, führt JSON eine Rekursion in die verschachtelten Objekte durch und beginnt mit demselben Untersuchungsdurchlauf. Wenn ein beliebiges Objekt eine `toJSON()`-Methode bereitstellt, verwendet JSON diese benutzerdefinierte Methode anstelle der internen Algorithmen.

Die JSON-Schnittstelle besteht aus einer Kodierungsmethode, `stringify()`, und einer Dekodierungsmethode, `parse()`. Beide Methoden stellen einen Parameter bereit, der es Ihnen erlaubt, Ihre eigene Logik in den JSON-Arbeitsablauf für das Kodieren oder Dekodieren einzufügen. Für `stringify()` heißt dieser Parameter `replacer`; für `parse()` ist es `reviver`. Diese Parameter nehmen eine Funktionsdefinition mit zwei Argumenten. Dazu wird die folgende Signatur verwendet:

```
function(k, v):*
```

toJSON()-Methoden

Die Signatur für `toJSON()`-Methoden lautet

```
public function toJSON(k:String):*
```

`JSON.stringify()` ruft `toJSON()`, sofern vorhanden, für jede öffentliche Eigenschaft auf, die bei der Untersuchung eines Objekts gefunden wird. Eine Eigenschaft besteht aus einem Schlüssel-Wert-Paar. Wenn `stringify()` `toJSON()` aufruft, wird der Schlüssel, `k`, der zurzeit untersuchten Eigenschaft übergeben. Eine typische `toJSON()`-Implementierung evaluiert jeden Eigenschaftennamen und gibt die gewünschte Kodierung ihres Werts zurück.

Die `toJSON()`-Methode kann Werte eines beliebigen Typs (gekennzeichnet als *) zurückgegeben, nicht nur Strings. Dieser variable Rückgabetyp ermöglicht `toJSON()`, ggf. ein Objekt zurückzugeben. Wenn zum Beispiel eine Eigenschaft in Ihrer benutzerdefinierten Klasse ein Objekt aus einer anderen Drittanbieterbibliothek enthält, können Sie dieses Objekt zurückgeben, wenn `toJSON()` auf die Eigenschaft trifft. JSON führt dann eine Rekursion in das Drittanbieterobjekt aus. Der Kodierungsprozessfluss verhält sich folgendermaßen:

- Wenn `toJSON()` ein Objekt zurückgibt, das nicht zu einem String evaluiert wird, führt `stringify()` eine Rekursion in dieses Objekt aus.
- Wenn `toJSON()` einen String zurückgibt, hüllt `stringify()` diesen in einen anderen String ein, gibt den umhüllten String zurück und fährt mit dem nächsten Wert fort.

Häufig ist es besser, ein Objekt zurückzugeben anstatt eines JSON-Strings, der von Ihrer Anwendung erstellt wurde. Durch die Zurückgabe eines Objekts wird der eingebaute JSON-Kodierungsalgorithmus genutzt, außerdem kann JSON dann eine Rekursion in verschachtelten Objekten ausführen.

Die `toJSON()`-Methode ist weder in der Object-Klasse noch in den meisten anderen nativen Klassen definiert. Das Nichtvorhandensein dieser Methode signalisiert JSON, den standardmäßigen Untersuchungsdurchlauf der öffentlichen Eigenschaften des Objekts auszuführen. Wenn Sie dies möchten, können Sie auch `toJSON()` verwenden, um die privaten Eigenschaften des Objekts bereitzustellen.

Einige native Klassen stellen jedoch Herausforderungen dar, die die ActionScript-Bibliotheken nicht für alle Verwendungszwecke lösen können. Für diese Klassen stellt ActionScript eine einfache Implementierung bereit, die Kunden für ihre Zwecke implementieren können. Die folgenden Klassen stellen triviale `toJSON()`-Mitglieder bereit:

- ByteArray
- Date
- Dictionary
- XML

Sie können eine Unterklasse der ByteArray-Klasse erstellen, um ihre `toJSON()`-Methode zu überschreiben, oder ihren Prototyp neu definieren. Da die Date- und XML-Klassen finale Klassen sind, müssen Sie den Klassenprototyp verwenden, um `toJSON()` neu zu definieren. Die Dictionary-Klasse ist dynamisch, was Ihnen zusätzliche Möglichkeiten beim Überschreiben von `toJSON()` gibt.

Definieren von JSON-Verhalten

Wenn Sie Ihre eigene JSON-Kodierung und Dekodierung für native Klassen implementieren möchten, haben Sie verschiedene Möglichkeiten:

- Definieren oder Überschreiben von `toJSON()` in Ihrer benutzerdefinierten Unterklasse einer nicht-finalen nativen Klasse
- Definieren oder Neudefinieren von `toJSON()` im Klassenprototyp
- Definieren einer `toJSON`-Eigenschaft in einer dynamischen Klasse
- Verwenden der Parameter `JSON.stringify() replacer` und `JSON.parser() reviver`

Verwandte Hilfethemen

ECMA-262, 5. Edition

Definieren von toJSON() für den Prototyp einer integrierten Klasse

Die native JSON-Implementierung in ActionScript spiegelt den ECMAScript-JSON-Mechanismus wider, der in ECMA-262, 5. Edition definiert ist. Da ECMAScript keine Klassen unterstützt, definiert ActionScript das JSON-Verhalten anhand prototyp-basierter Absetzungen. Prototypen sind Vorläufer von ActionScript 3.0-Klassen, die das simulierte Vererben sowie das Hinzufügen und Neudefinieren von Mitgliedern zulassen.

Mit ActionScript können Sie `toJSON()` für den Prototyp jeder Klasse definieren oder neu definieren. Dies gilt sogar für Klassen, die als final gekennzeichnet sind. Wenn Sie `toJSON()` für einen Klassenprototyp definieren, wird Ihre Definition die aktuelle Definition für alle Instanzen dieser Klasse im Rahmen Ihrer Anwendung. So können Sie zum Beispiel eine `toJSON()`-Methode für den MovieClip-Prototyp definieren:

```
MovieClip.prototype.toJSON = function(k):* {
    trace("prototype.toJSON() called.");
    return "toJSON";
}
```

Wenn Ihre Anwendung dann `stringify()` für eine beliebige MovieClip-Instanz aufruft, gibt `stringify()` die Ausgabe Ihrer `toJSON()`-Methode zurück:

```
var mc:MovieClip = new MovieClip();
var js:String = JSON.stringify(mc); //"prototype toJSON() called."
trace("js: " + js); //"js: toJSON"
```

Sie können `toJSON()` auch in nativen Klassen, die diese Methode definieren, überschreiben. Mit dem folgenden Code überschreiben Sie zum Beispiel `Date.toJSON()`:

```
Date.prototype.toJSON = function (k):* {
    return "any date format you like via toJSON: "+
        "this.time:"+this.time + " this.hours:"+this.hours;
}
var dt:Date = new Date();
trace(JSON.stringify(dt));
// "any date format you like via toJSON: this.time:1317244361947 this.hours:14"
```

Definieren oder Überschreiben von toJSON() auf Klassenebene

Anwendungen müssen nicht immer Prototypen verwenden, um `toJSON()` neu zu definieren. Sie können `toJSON()` auch als Mitglied einer Unterklasse definieren, falls die übergeordnete Klasse nicht als finale Klasse gekennzeichnet ist. Sie können zum Beispiel die ByteArray-Klasse erweitern und eine öffentliche `toJSON()`-Funktion definieren:

```
package {

    import flash.utils.ByteArray;
    public class MyByteArray extends ByteArray
    {
        public function MyByteArray() {
        }

        public function toJSON(s:String):*
        {
            return "MyByteArray";
        }

    }
}


var ba:ByteArray = new ByteArray();
trace(JSON.stringify(ba)); //"ByteArray"
var mba:MyByteArray = new MyByteArray(); //"MyByteArray"
trace(JSON.stringify(mba)); //"MyByteArray"
```

Wenn eine Klasse dynamisch ist, können Sie einem Objekt dieser Klasse eine `toJSON`-Eigenschaft hinzufügen und ihr auf folgende Weise eine Funktion zuweisen:

```
var d:Dictionary = new Dictionary();
trace(JSON.stringify((d))); // "Dictionary"
d.toJSON = function(){return {c : "toJSON override."};} // overrides existing function
trace(JSON.stringify((d))); // {"c":"toJSON override."}
```

Sie können `toJSON()` in jeder ActionScript-Klasse überschreiben, definieren oder neu definieren. Die meisten integrierten ActionScript-Klassen definieren `toJSON()` jedoch nicht. Die Object-Klasse definiert `toJSON` weder in ihrem Standardprototyp noch als Klassenmitglied. Nur wenige native Klassen definieren die Methode als Prototypfunktion. Deshalb können Sie `toJSON()` in den meisten Klassen nicht im herkömmlichen Sinne überschreiben.

Native Klassen, die `toJSON()` nicht definieren, werden durch die interne JSON-Implementierung zu JSON serialisiert. Vermeiden Sie nach Möglichkeit, diese eingebaute Funktionalität zu ersetzen. Wenn Sie ein `toJSON()`-Mitglied definieren, verwendet die JSON-Klasse statt ihrer eigenen Funktionalität Ihre Logik.

Verwenden des replacer-Parameters von JSON.stringify()

Das Überschreiben von `toJSON()` im Prototyp ist hilfreich zum Ändern des JSON-Exportverhaltens einer Klasse in der gesamten Anwendung. In einigen Fällen gilt Ihre Exportlogik aber möglicherweise nur für bestimmte Szenarien unter vorübergehenden Bedingungen. Um diese Änderungen mit kleinem Umfang zu ermöglichen, können Sie den `replacer`-Parameter der `JSON.stringify()`-Methode verwenden.

Die `stringify()`-Methode wendet die über den `replacer`-Parameter übergebene Funktion auf das Objekt an, das kodiert wird. Die Signatur dieser Funktion ähnelt der von `toJSON()`:

```
function (k,v):*
```

Anders als `toJSON()` erfordert die `replacer`-Funktion sowohl den Wert, `v`, als auch den Schlüssel `k`. Dieser Unterschied ist notwendig, da `stringify()` für das statische JSON-Objekt anstatt für das kodierte Objekt definiert wird. Wenn `JSON.stringify()replacer(k,v)` aufruft, wird das ursprüngliche Eingabeobjekt untersucht. Der implizite `this`-Parameter, der an die `replacer`-Funktion übergeben wird, verweist auf das Objekt, das den Schlüssel und den Wert enthält. Da `JSON.stringify()` das ursprüngliche Eingabeobjekt nicht verändert, bleibt dieses Objekt unverändert im untersuchten Container. Sie können den Code `this[k]` verwenden, um den Schlüssel im ursprünglichen Objekt abzufragen. Der `v`-Parameter enthält den Wert, den `toJSON()` konvertiert.

Wie auch `toJSON()` kann die `replacer`-Funktion Werte beliebiger Typen ausgeben. Wenn `replacer` einen String zurückgibt, setzt die JSON-Engine die Inhalte in Anführungszeichen und umhüllt diese geschützten Inhalte dann ebenfalls mit Anführungszeichen. Dieses Umhüllen stellt sicher, dass `stringify()` ein gültiges JSON-Stringobjekt empfängt, das in einem nachfolgenden Aufruf von `JSON.parse()` auch ein String bleibt.

Der folgende Code verwendet den `replacer`-Parameter und den impliziten `this`-Parameter, um die `time`- und `hours`-Werte eines Date-Objekts zurückzugeben:

```
JSON.stringify(d, function (k,v):* {
    return "any date format you like via replacer: "+
        "holder[k].time:"+this[k].time + " holder[k].hours:"+this[k].hours;
});
```

Verwenden des reviver-Parameters von JSON.parse()

Der `reviver`-Parameter der `JSON.parse()`-Method ist das Gegenstück zur `replacer`-Funktion: Er konvertiert einen JSON-String in ein verwendbares ActionScript-Objekt. Das `reviver`-Argument ist eine Funktion, die zwei Parameter nimmt und einen beliebigen Typ zurückgeben kann:

```
function (k,v):*
```

In dieser Funktion ist `k` ein Schlüssel und `v` ist der Wert von `k`. Wie `stringify()`, so untersucht `parse()` die JSON-Schlüssel-Wert-Paare und wendet die `reviver`-Funktion, sofern vorhanden, auf jedes Paar an. Ein potenzielles Problem ist die Tatsache, dass die JSON-Klasse nicht den ActionScript-Klassennamen eines Objekts ausgibt. Deshalb kann es schwierig sein, zu wissen, auf welches Objekt die reviver-Funktion angewendet werden soll. Dies gilt umso mehr, wenn Objekte verschachtelt sind. Beim Entwurf von `toJSON()`, `replacer`- und `reviver`-Funktionen können Sie Möglichkeiten einplanen, die ActionScript-Objekte zu identifizieren, die exportiert werden, während die Originalobjekte intakt bleiben.

Parsing-Beispiel

Das folgende Beispiel demonstriert eine Strategie zum Anwenden der reviver-Funktion auf Objekte, die aus JSON-Strings geparst wurden. In diesem Beispiel werden zwei Klassen definiert: JSONGenericDictExample und JSONDictionaryExtnExample. Die JSONGenericDictExample-Klasse ist eine benutzerdefinierte Wörterbuchklasse. Jeder Datensatz enthält den Namen und den Geburtstag einer Person sowie eine eindeutige ID. Jedes Mal, wenn der JSONGenericDictExample-Konstruktor aufgerufen wird, fügt er das neu erstellte Objekt einem internen statischen Array hinzu, mit einer statisch inkrementierten Ganzzahl als ID. Die JSONGenericDictExample-Klasse definiert auch eine `revive()`-Methode, die nur den Ganzzahlteil aus dem längeren `id`-Mitglied extrahiert. Die `revive()`-Methode verwendet diese Ganzzahl, um das richtige Objekt nachzuschlagen und zurückzugeben.

Die JSONDictionaryExtnExample-Klasse erweitert die ActionScript-Dictionary-Klasse. Ihre Datensätze haben keine festgelegte Struktur und können beliebige Daten enthalten. Daten werden nach der Konstruktion eines JSONDictionaryExtnExample-Objekts zugewiesen, nicht als klassendefinierte Eigenschaften. JSONDictionaryExtnExample-Datensätze verwenden JSONGenericDictExample-Objekte als Schlüssel. Wenn ein JSONDictionaryExtnExample-Objekt wiederbelebt wird, verwendet die `JSONGenericDictExample.revive()`-Funktion die ID, die JSONDictionaryExtnExample zugeordnet ist, um das richtige Schlüsselobjekt abzurufen.

Am wichtigsten ist, dass die `JSONDictionaryExtnExample.toJSON()`-Methode zusätzlich zum JSONDictionaryExtnExample-Objekt einen Markerstring zurückgibt. Dieser String kennzeichnet die JSON-Ausgabe als zur JSONDictionaryExtnExample-Klasse gehörig. Dieser Marker gibt eindeutig an, welcher Objekttyp während `JSON.parse()` verarbeitet wird.

```
package {
    // Generic dictionary example:
    public class JSONGenericDictExample {
        static var revivableObjects = [];
        static var nextId = 10000;
        public var id;
        public var dname:String;
        public var birthday;

        public function JSONGenericDictExample(name, birthday) {
            revivableObjects[nextId] = this;
            this.id       = "id_class_JSONGenericDictExample_" + nextId;
            this.dname     = name;
            this.birthday = birthday;
            nextId++;
        }
        public function toString():String { return this.dname; }
        public static function revive(id:String):JSONGenericDictExample {
            var r:RegExp = /^id_class_JSONGenericDictExample_([0-9]*)$/;
            var res = r.exec(id);
            return JSONGenericDictExample.revivableObjects[res[1]];
        }
    }
}

package {
    import flash.utils.Dictionary;
    import flash.utils.ByteArray;

    // For this extension of dictionary, we serialize the contents of the
    // dictionary by using toJSON
    public final class JSONDictionaryExtnExample extends Dictionary {
        public function toJSON(k):* {
            var contents = {};
            for (var a in this) {
                contents[a.id] = this[a];
            }

            // We also wrap the contents in an object so that we can
            // identify it by looking for the marking property "class E"
            // while in the midst of JSON.parse.
```

```
            return {"class JSONDictionaryExtnExample": contents};
        }

        // This is just here for debugging and for illustration
        public function toString():String {
            var retval = "[JSONDictionaryExtnExample <";
            var printed_any = false;
            for (var k in this) {
                retval += k.toString() + "=" +
                "[e="+this[k].earnings +
                ",v="+this[k].violations + "], "
                printed_any = true;
            }
            if (printed_any)
                retval = retval.substring(0, retval.length-2);
            retval += ">]"
            return retval;
        }
    }
}
```

Wenn das folgende Laufzeitskript `JSON.parse()` für ein JSONDictionaryExtnExample-Objekt aufruft, ruft die `reviver`-Funktion `JSONGenericDictExample.revive()` für jedes Objekt in JSONDictionaryExtnExample auf. Mit diesem Aufruf wird die ID extrahiert, die den Objektschlüssel darstellt. Die `JSONGenericDictExample.revive()`-Funktion verwendet diese ID, um das gespeicherte JSONDictionaryExtnExample-Objekt aus einem privaten statischen Array abzurufen und zurückzugeben.

```
import flash.display.MovieClip;
import flash.text.TextField;

var a_bob1:JSONGenericDictExample = new JSONGenericDictExample("Bob", new
Date(Date.parse("01/02/1934")));
var a_bob2:JSONGenericDictExample = new JSONGenericDictExample("Bob", new
Date(Date.parse("05/06/1978")));
var a_jen:JSONGenericDictExample = new JSONGenericDictExample("Jen", new
Date(Date.parse("09/09/1999")));

var e = new JSONDictionaryExtnExample();
e[a_bob1] = {earnings: 40, violations: 2};
e[a_bob2] = {earnings: 10, violations: 1};
e[a_jen]  = {earnings: 25, violations: 3};

trace("JSON.stringify(e): " + JSON.stringify(e)); // {"class JSONDictionaryExtnExample":
                        //{"id_class_JSONGenericDictExample_10001":
                        //{"earnings":10,"violations":1},
                        //"id_class_JSONGenericDictExample_10002":
                        //{"earnings":25,"violations":3},
                        //"id_class_JSONGenericDictExample_10000":
                        // {"earnings":40,"violations":2}}}

var e_result = JSON.stringify(e);

var e1 = new JSONDictionaryExtnExample();
var e2 = new JSONDictionaryExtnExample();

// It's somewhat easy to convert the string from JSON.stringify(e) back
// into a dictionary (turn it into an object via JSON.parse, then loop
```

```
// over that object's properties to construct a fresh dictionary).
//
// The harder exercise is to handle situations where the dictionaries
// themselves are nested in the object passed to JSON.stringify and
// thus does not occur at the topmost level of the resulting string.
//
// (For example: consider roundtripping something like
//   var tricky_array = [e1, [[4, e2, 6]], {table:e3}]
// where e1, e2, e3 are all dictionaries.  Furthermore, consider
// dictionaries that contain references to dictionaries.)
//
// This parsing (or at least some instances of it) can be done via
// JSON.parse, but it's not necessarily trivial.  Careful consideration
// of how toJSON, replacer, and reviver can work together is
// necessary.

var e_roundtrip =
    JSON.parse(e_result,
            // This is a reviver that is focused on rebuilding JSONDictionaryExtnExample objects.
                function (k, v) {
                     if ("class JSONDictionaryExtnExample" in v) { // special marker tag;
                         //see JSONDictionaryExtnExample.toJSON().
                        var e = new JSONDictionaryExtnExample();
                        var contents = v["class JSONDictionaryExtnExample"];
                        for (var i in contents) {
                           // Reviving JSONGenericDictExample objects from string
                           // identifiers is also special;
                           // see JSONGenericDictExample constructor and
                           // JSONGenericDictExample's revive() method.
                            e[JSONGenericDictExample.revive(i)] = contents[i];
                        }
                        return e;
                    } else {
                        return v;
                    }
                });

trace("// == Here is an extended Dictionary that has been round-tripped  ==");
trace("// == Note that we have revived Jen/Jan during the roundtrip.    ==");
trace("e:           " + e); //[JSONDictionaryExtnExample <Bob=[e=40,v=2], Bob=[e=10,v=1],
                            //Jen=[e=25,v=3]>]
trace("e_roundtrip: " + e_roundtrip); //[JSONDictionaryExtnExample <Bob=[e=40,v=2],
                                      //Bob=[e=10,v=1], Jen=[e=25,v=3]>]
trace("Is e_roundtrip a JSONDictionaryExtnExample? " + (e_roundtrip is
JSONDictionaryExtnExample)); //true
trace("Name change: Jen is now Jan");
a_jen.dname = "Jan"

trace("e:           " + e); //[JSONDictionaryExtnExample <Bob=[e=40,v=2], Bob=[e=10,v=1],
                            //Jan=[e=25,v=3]>]
trace("e_roundtrip: " + e_roundtrip); //[JSONDictionaryExtnExample <Bob=[e=40,v=2],
                                      //Bob=[e=10,v=1], Jan=[e=25,v=3]>]
```

Kapitel 8: Verarbeiten von Ereignissen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Ereignis verarbeitendes System bietet Programmierern eine praktische Möglichkeit, auf Benutzereingaben und Systemereignisse zu reagieren. Das Ereignismodell von ActionScript 3.0 ist nicht nur bequem, sondern auch standardkonform und gut in die Anzeigeliste integriert. Das neue Ereignismodell beruht auf der DOM3-Ereignisspezifikation (Document Object Model Level 3), eine dem Branchenstandard entsprechende Ereignisverarbeitungsarchitektur. Es ist ein leistungsfähiges und dabei intuitiv verständliches Werkzeug zur Ereignisverarbeitung für ActionScript-Programmierer.

Das ActionScript 3.0-Ereignisverarbeitungssystem arbeitet eng mit der Anzeigeliste zusammen. Grundlegende Informationen zur Anzeigeliste finden Sie unter „Programmieren von Anzeigeobjekten“ auf Seite 161.

Verwandte Hilfethemen

flash.events-Paket

DOM3-Ereignisspezifikation (Document Object Model Level 3)

Grundlagen der Ereignisverarbeitung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können sich Ereignisse (Englisch „events“) als beliebige Phänomene in SWF-Dateien vorstellen, die für Sie als Programmierer von Interesse sind. Beispielsweise unterstützen die meisten SWF-Dateien eine gewisse Interaktion mit den Benutzern. Dabei kann es sich um etwas so Einfaches wie die Reaktion auf Mausklicks oder um etwas Komplexeres wie die Eingabe und Verarbeitung von Formulardaten handeln. Jede dieser Interaktionen mit einer SWF-Datei stellt ein Ereignis dar. Ereignisse können auch ohne direkte Benutzerinteraktion auftreten, beispielsweise wenn Daten vollständig von einem Server heruntergeladen wurden oder eine angeschlossene Kamera aktiviert wurde.

In ActionScript 3.0 werden alle Ereignisse durch ein Ereignisobjekt dargestellt, das eine Instanz der Event-Klasse oder eine ihrer Unterklassen ist. In einem Ereignisobjekt werden nicht nur Informationen zu einem bestimmten Ereignis gespeichert, sondern es enthält auch Methoden, die die Bearbeitung des Ereignisobjekts vereinfachen. Wenn in Flash Player oder AIR beispielsweise ein Mausklick erkannt wird, wird ein Ereignisobjekt (eine Instanz der MouseEvent-Klasse) angelegt, um genau dieses Mausklickereignis abzubilden.

Nach dem Erstellen eines Ereignisobjekts wird dieses in Flash Player oder AIR *ausgelöst*. Das heißt, das Ereignisobjekt wird an das Zielobjekt für das Ereignis übergeben. Ein Objekt, das als Ziel für ein ausgelöstes Ereignisobjekt dient, wird als *Ereignisziel* bezeichnet. Wenn beispielsweise eine angeschlossene Kamera aktiviert wird, löst Flash Player ein Ereignisobjekt aus, das direkt an das Ereignisziel übergeben wird – in diesem Fall an das Objekt, das die Kamera repräsentiert. Wenn sich das Ereignisziel jedoch in der Anzeigeliste befindet, wird das Ereignisobjekt durch die Hierarchie der Anzeigeliste nach unten weitergeleitet, bis das Ereignisziel erreicht ist. In einigen Fällen bewegt sich das Ereignisobjekt anschließend auf demselben Weg in der Hierarchie der Anzeigeliste wieder nach oben. Dieses Durchlaufen der Hierarchie in der Anzeigeliste wird als *Ereignisablauf* bezeichnet.

Sie können mit dem Code auf Ereignisse „warten“, indem Sie Ereignis-Listener verwenden. *Ereignis-Listener* (Englisch „event listener“) sind programmierte Funktionen oder Methoden, mit denen auf bestimmte Ereignisse reagiert werden kann. Um sicherzustellen, dass ein Programm auf Ereignisse reagiert, müssen Sie dem Ereignisziel oder einem Anzeigelistenobjekt, das Bestandteil des Ereignisablaufs eines Ereignisobjekts ist, Ereignis-Listener hinzufügen.

Jeder als Ereignis-Listener erstellte Code hat die folgende Grundstruktur (Elemente in Fettdruck sind Platzhalter, die Sie je nach Zweck des Programms ersetzen):

```
function eventResponse(eventObject:EventType):void
{
    // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

Dieser Code bewirkt zweierlei. Zunächst wird eine Funktion definiert. Auf diese Weise werden die Aktionen festgelegt, die als Reaktion auf das Ereignis ausgeführt werden. Dann wird die `addEventListener()`-Methode des Quellobjekts aufgerufen. Dabei wird die Funktion für das Quellobjekt „registriert“, damit die Aktionen der Funktion beim Eintreten des Ereignisses ausgeführt werden. Beim Eintreten des Ereignisses wird dann für das Ereignisziel die Liste aller als Ereignis-Listener registrierten Funktionen und Methoden überprüft. Danach werden alle Funktionen und Methoden aufgerufen, wobei das Ereignisobjekt als Parameter übergeben wird.

Um einen benutzerdefinierten Ereignis-Listener zu erstellen, müssen Sie den Code an vier Stellen ändern. Ändern Sie zunächst den Namen der Funktion wie gewünscht (an zwei Stellen, an denen im Code **eventResponse** steht). Geben Sie dann den entsprechenden Klassennamen des Ereignisobjekts an, das von dem Ereignis ausgelöst wird, für das der Listener bestimmt ist (im Code **EventType**). Legen Sie die entsprechende Konstante für das jeweilige Ereignis fest (im Listing **EVENT_NAME**). Zum Schluss müssen Sie die `addEventListener()`-Methode für das Objekt aufrufen, das das Ereignis auslöst (in diesem Code **eventTarget**). Optional können Sie den Namen der als Funktionsparameter verwendeten Variablen ändern (im Code **eventObject**).

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die Ihnen beim Schreiben von Routinen zur Ereignisverarbeitung häufig begegnen:

Bubbling Bubbling tritt bei einigen Ereignissen ein, damit ein übergeordnetes Anzeigeobjekt auf Ereignisse reagieren kann, die von seinen untergeordneten Objekten ausgelöst wurden.

Bubbling-Phase (Aufstiegsphase) Der Teil eines Ereignisablaufs, in dem ein Ereignis zum übergeordneten Anzeigeobjekt „aufsteigt“. Die Bubbling-Phase erfolgt nach den Erfassungs- und Zielphasen.

Erfassungsphase Der Teil des Ereignisablaufs, in dem ein Ereignis vom allgemeinsten Ziel zum spezifischsten Zielobjekt „absteigt“. Die Erfassungsphase erfolgt vor den Ziel- und Bubbling-Phasen.

Standardverhalten Einige Ereignisse verfügen über ein Verhalten, das normalerweise zusammen mit dem Ereignis auftritt. Dies wird als Standardverhalten bezeichnet. Wenn ein Benutzer beispielsweise Text in einem Textfeld eingibt, wird ein Texteingabeereignis ausgelöst. Das Standardverhalten für dieses Ereignis ist das Anzeigen des im Textfeld eingegebenen Zeichens. Sie können dieses Standardverhalten jedoch außer Kraft setzen (wenn beispielsweise die eingegebenen Zeichen nicht angezeigt werden sollen).

Auslösen Das Benachrichtigen von Ereignis-Listenern, dass ein Ereignis aufgetreten ist.

Ereignis Ein Ereignis, das für ein Objekt auftritt und über das dieses Objekt andere Objekte benachrichtigen kann.

Ereignisablauf Wenn Ereignisse für ein Objekt in der Anzeigeliste (ein auf dem Bildschirm angezeigtes Objekt) auftreten, werden alle Objekte, die dieses Objekt enthalten, über das Ereignis benachrichtigt und benachrichtigen dann jeweils die zugehörigen Ereignis-Listener. Dieser Vorgang beginnt bei der Bühne, setzt sich durch die

Anzeigeliste bis zu dem Objekt fort, für das das Ereignis aufgetreten ist, und kehrt dann wieder zur Bühne zurück. Dies wird als Ereignisablauf bezeichnet.

Ereignisobjekt Ein Objekt, das Informationen über das Auftreten eines bestimmten Ereignisses enthält und beim Auslösen des Ereignisses an alle Listener gesendet wird.

Ereigniszieł Das Objekt, das ein Ereignis auslöst. Wenn der Benutzer beispielsweise auf eine Schaltfläche innerhalb einer Sprite-Instanz innerhalb der Bühne klickt, lösen alle diese Objekte Ereignisse aus. Das Ereigniszieł ist jedoch das Objekt, bei dem das Ereignis tatsächlich aufgetreten ist – in diesem Fall die Schaltfläche, auf die geklickt wurde.

Listener Objekte oder Funktionen, die sich selbst bei einem Objekt registriert haben, um anzuzeigen, dass sie beim Auftreten eines bestimmten Ereignisses benachrichtigt werden sollen.

Zielphase Der Punkt im Ereignisablauf, an dem das Ereignis das spezifischste Ziel erreicht hat. Die Zielphase erfolgt zwischen der Erfassungs- und der Bubbling-Phase.

Unterschiede der Ereignisverarbeitung in ActionScript 3.0 im Vergleich mit früheren Versionen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der deutlichste Unterschied zwischen der Ereignisverarbeitung in ActionScript3 .0 und der in älteren ActionScript-Versionen besteht darin, dass es in ActionScript 3.0 nur ein System der Ereignisverarbeitung gibt, während die bisherigen ActionScript-Versionen über mehrere unterschiedliche Ereignisverarbeitungssysteme verfügten. Dieser Abschnitt beginnt mit einem Überblick über die Ereignisverarbeitung in früheren ActionScript-Versionen und behandelt dann die Änderungen in ActionScript 3.0.

Ereignisverarbeitung in früheren ActionScript-Versionen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In den ActionScript-Versionen vor ActionScript 3.0 gab es eine Reihe unterschiedlicher Arten der Ereignisverarbeitung:

- `on()`-Ereignisprozeduren, die direkt für Button- und MovieClip-Instanzen definiert werden können
- `onClipEvent()`-Ereignisprozeduren, die direkt für MovieClip-Instanzen definiert werden können
- Eigenschaften von Callback-Funktionen wie `XML.onload` oder `Camera.onActivity`
- Ereignis-Listener, die mit der `addListener()`-Methode registriert werden
- Die UIEventDispatcher-Klasse, mit der das DOM-Ereignismodell teilweise implementiert wurde

Jeder dieser Mechanismen bietet jeweils Vorteile, hat jedoch auch Beschränkungen. Die Ereignisprozeduren `on()` und `onClipEvent()` sind einfach einzusetzen, können die spätere Pflege von Projekten jedoch erschweren, da direkt für Schaltflächen und Movieclips definierter Code unter Umständen schwer zu finden ist. Callback-Funktionen können ebenfalls einfach implementiert werden, sind jedoch auf jeweils eine Funktion pro Ereignis beschränkt. Die Implementierung von Ereignis-Listenern ist schwieriger. Es müssen nicht nur ein Listener-Objekt und eine Listener-Funktion erstellt werden, sondern der Listener muss auch in dem Objekt registriert werden, mit dem das Ereignis generiert wird. Durch diesen erhöhten Verwaltungsaufwand können Sie jedoch mehrere Listener-Objekte erstellen und alle für dasselbe Ereignis registrieren.

Die Entwicklung von Komponenten für ActionScript 2.0 brachte ein weiteres Ereignismodell mit sich. Dieses neue Modell, verkörpert durch die UIEventDispatcher-Klasse, basierte auf einem Teilbereich der DOM-Ereignisspezifikation. Für Entwickler, die mit der Komponentenereignisverarbeitung vertraut sind, gestaltet sich der Übergang zum neuen Ereignismodell von ActionScript 3.0 relativ problemlos.

Leider gibt es zahlreiche Überschneidungen bei der in den verschiedenen Ereignismodellen verwendeten Syntax sowie einige Abweichungen. Beispielsweise können in ActionScript 2.0 einige Eigenschaften wie etwa `TextField.onChanged` entweder als Callback-Funktion oder als Ereignis-Listener verwendet werden. Die Syntax zum Registrieren von Listener-Objekten unterscheidet sich jedoch in Abhängigkeit davon, ob Sie die UIEventDispatcher-Klasse oder eine der sechs Klassen verwenden, die Listener unterstützen. Für die Key-, Mouse-, MovieClipLoader-, Selection-, Stage- und TextField-Klassen wird die `addListener()`-Methode verwendet, für die Komponentenereignisverarbeitung jedoch die `addEventListener()`-Methode.

Eine weitere durch die unterschiedlichen Ereignisverarbeitungsmodelle hervorgerufene Schwierigkeit liegt darin, dass der Gültigkeitsbereich der Ereignisprozedurfunktion je nach verwendetem Mechanismus große Unterschiede aufweist. Mit anderen Worten, die Bedeutung des Schlüsselworts `this` ist in den jeweiligen Ereignisverarbeitungssystemen nicht konsistent.

Ereignisverarbeitung in ActionScript 3.0

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit ActionScript 3.0 wird ein einziges Ereignisverarbeitungsmodell eingeführt, das die vielen unterschiedlichen Ereignisverarbeitungsmechanismen der Vorgängerversionen ersetzt. Das neue Ereignismodell beruht auf der DOM3-Ereignisspezifikation (Document Object Model Level 3). Obwohl sich das SWF-Dateiformat nicht speziell an den DOM-Standard hält, liegen genug Ähnlichkeiten zwischen der Anzeigeliste und der DOM-Struktur vor, um die Implementierung des DOM-Ereignismodells zu ermöglichen. Ein Objekt in der Anzeigeliste entspricht einem Knoten in der hierarchischen DOM-Struktur. Die Begriffe *Anzeigelistenobjekt* und *Knoten* werden im Folgenden synonym verwendet.

Die Flash Player- und AIR-Implementierung des DOM-Ereignismodells enthält ein neues Konzept: das Standardverhalten. Ein *Standardverhalten* ist eine Aktion, die in Flash Player bzw. AIR als normale Reaktion auf bestimmte Ereignisse ausgeführt wird.

Standardverhalten

Normalerweise sind Entwickler dafür zuständig, Code zu schreiben, mit dem auf Ereignisse reagiert wird. In einigen Fällen ist die Zuordnung eines Verhaltens zu einem Ereignis jedoch so üblich, dass Flash Player bzw. AIR das Verhalten automatisch ausführt, wenn der Entwickler nicht Code hinzufügt, um dies zu verhindern. Da das Ausführen dieser Verhalten in Flash Player bzw. AIR automatisch erfolgt, werden diese als Standardverhalten bezeichnet.

Wenn ein Benutzer beispielsweise Text in ein TextField-Objekt eingibt, wird in der Regel davon ausgegangen, dass dieser Text im TextField-Objekt angezeigt wird. Deshalb ist dieses Verhalten in Flash Player und AIR integriert. Wenn dieses Standardverhalten nicht auftreten soll, können Sie es mithilfe des neuen Ereignisverarbeitungssystems unterdrücken. Wenn Benutzer Text in ein TextField-Objekt eingeben, wird in Flash Player bzw. AIR eine Instanz der TextEvent-Klasse erstellt, die diese Benutzereingabe repräsentiert. Um zu verhindern, dass Flash Player bzw. AIR den Text im TextField-Objekt anzeigt, müssen Sie auf diese TextEvent-Instanz zugreifen und die entsprechende `preventDefault()`-Methode aufrufen.

Es können jedoch nicht alle Standardverhalten verhindert werden. Beispielsweise erzeugen Flash Player und AIR ein MouseEvent-Objekt, wenn ein Benutzer auf ein Wort in einem TextField-Objekt doppelklickt. Durch das Standardverhalten, das nicht unterdrückt werden kann, wird das Wort unter dem Mauszeiger markiert.

Vielen Ereignisobjekten ist kein Standardverhalten zugewiesen. Beispielsweise löst Flash Player ein Verbindungsereignis aus, wenn eine Netzwerkverbindung hergestellt wird, es gibt jedoch kein entsprechendes Standardverhalten. In der API-Dokumentation für die Event-Klasse und ihre Unterklassen sind alle Ereignistypen aufgeführt. Hier finden Sie alle zugewiesenen Standardverhalten sowie die Angabe, ob dieses Verhalten verhindert werden kann.

Es ist wichtig zu verstehen, dass Standardverhalten nur Ereignisobjekten zugeordnet sind, die von Flash Player bzw. AIR ausgelöst wurden. Für im ActionScript-Programmcode ausgelöste Ereignisobjekte sind keine Standardverhalten vorhanden. Sie können beispielsweise die Methoden der EventDispatcher-Klasse verwenden, um ein Ereignisobjekt vom Typ `textInput` auszulösen. Diesem Ereignisobjekt ist jedoch kein Standardverhalten zugeordnet. Das bedeutet, dass Flash Player udn AIR als Ergebnis eines vom Programmcode ausgelösten `textInput`-Ereignisses keine Zeichen in einem TextField-Objekt anzeigen.

Neues bei Ereignis-Listenern in ActionScript 3.0

Für Entwickler, die bereits Erfahrungen mit der `addListener()`-Methode in ActionScript 2.0 gesammelt haben, ist eine Übersicht der Unterschiede zwischen dem Ereignis-Listener-Modell von ActionScript 2.0 und dem Ereignismodell von ActionScript 3.0 hilfreich. Im Folgenden sind daher einige der wichtigsten Unterschiede zwischen den beiden Ereignismodellen aufgeführt:

- Zum Hinzufügen von Ereignis-Listenern in ActionScript 2.0 wird in einigen Fällen `addListener()` und in anderen `addEventListener()` verwendet. In ActionScript 3.0 wird hingegen in allen Fällen `addEventListener()` eingesetzt.
- In ActionScrip 2.0 gibt es keinen Ereignisablauf, d. h. die `addListener()`-Methode kann nur für das Objekt aufgerufen werden, das das Ereignis überträgt. Im Gegensatz dazu kann in ActionScript 3.0 die `addEventListener()`-Methode für jedes Objekt aufgerufen werden, das Bestandteil des Ereignisablaufs ist.
- In ActionScript 2.0 können Ereignis-Listener entweder Funktionen, Methoden oder Objekte sein, während in ActionScript 3.0 als Ereignis-Listener nur Funktionen oder Methoden möglich sind.

Der Ereignisablauf

Flash Player 9 und höher, Adobe AIR 1.0 und höher

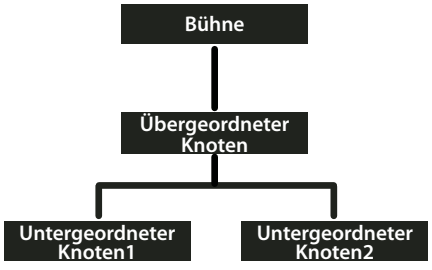
Beim Auftreten von Ereignissen löst Flash Player bzw. AIR Ereignisobjekte aus. Wenn sich das Ereignisziel nicht in der Anzeigeliste befindet, sendet Flash Player bzw. AIR das Ereignisobjekt direkt an das Ereignisziel. Beispielsweise sendet Flash Player das Fortschrittsereignisobjekt direkt an ein URLStream-Objekt. Wenn sich das Ereignisziel jedoch in der Anzeigeliste befindet, sendet Flash Player das Ereignisobjekt an die Anzeigeliste. Das Ereignisobjekt durchläuft dann die Anzeigeliste, bis das Ereignisziel erreicht ist.

Mit dem *Ereignisablauf* wird die Bewegung eines Ereignisobjekts innerhalb der Anzeigeliste beschrieben. Die Anzeigeliste ist in einer Hierarchie angeordnet, die als Baumstruktur bezeichnet werden kann. An oberster Stelle der Anzeigelistenhierarchie befindet sich die Bühne, ein spezieller Anzeigeobjektcontainer, der als Ausgangspunkt der Anzeigeliste dient. Die Bühne wird durch die flash.display.Stage-Klasse repräsentiert. Auf sie kann nur über ein Anzeigeobjekt zugegriffen werden. Jedes Anzeigeobjekt verfügt über eine Eigenschaft mit dem Namen `stage`, die sich auf die Bühne der jeweiligen Anwendung bezieht.

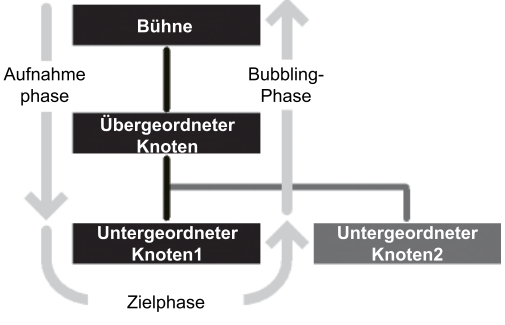
Wenn Flash Player oder AIR ein Ereignisobjekt für ein Ereignis im Zusammenhang mit einer Anzeigeliste auslöst, durchläuft dieses Ereignisobjekt die Hierarchie von der Bühne bis zum *Zielknoten* und zurück. Nach der DOM-Ereignisspezifikation ist der Zielknoten als der Knoten definiert, der das Ereignisziel repräsentiert. Anders ausgedrückt, ist der Zielknoten das Anzeigelistenobjekt, in dem das Ereignis aufgetreten ist. Wenn ein Benutzer beispielsweise auf ein Anzeigelistenobjekt mit dem Namen `Untergeordneter Knoten 1` klickt, löst Flash Player bzw. AIR ein Ereignisobjekt aus und verwendet `Untergeordneter Knoten 1` als Zielknoten.

Der Ereignisablauf ist konzeptionell in drei Abschnitte unterteilt. Der erste Abschnitt wird als Empfangsphase bezeichnet. Diese Phase besteht aus allen Knoten von der Bühne bis zum übergeordneten Knoten des Zielknotens. Der zweite Abschnitt ist die sogenannte Zielphase, die nur aus dem Zielknoten besteht. Der dritte Abschnitt wird Aufstiegsphase genannt. Diese besteht aus den Knoten, die vom übergeordneten Knoten des Zielknotens bis zur Bühne zurück durchlaufen werden.

Die Namen der einzelnen Phasen ergeben mehr Sinn, wenn Sie sich die Anzeigeliste als vertikale Hierarchie mit der Bühne an oberster Stelle vorstellen, wie in der folgenden Abbildung dargestellt:



Wenn ein Benutzer auf `Untergeordneter Knoten 1` klickt, löst Flash Player bzw. AIR ein Ereignisobjekt aus und übergibt es an den Ereignisablauf. Wie in der folgenden Abbildung dargestellt ist, startet das Objekt bei `Bühne`, bewegt sich nach unten zu `Übergeordneter Knoten`, dann zu `Untergeordneter Knoten 1` und bewegt sich dann wieder nach oben bis zu `Bühne`. Dabei wird `Übergeordneter Knoten` auf dem Weg zu `Bühne` erneut durchlaufen.



In diesem Beispiel umfasst die Empfangsphase `Bühne` und `Übergeordneter Knoten` während der anfänglichen Abwärtsbewegung. Die Zielphase ist der Zeitraum, in dem sich das Objekt in `Untergeordneter Knoten 1` befindet. Die Aufstiegsphase umfasst `Übergeordneter Knoten` und `Bühne`, die bei der Aufwärtsbewegung zum Stammknoten durchlaufen werden.

Der Ereignisablauf trägt zu einem Ereignisverarbeitungssystem bei, das leistungsfähiger ist als die Ereignisverarbeitung, die ActionScript-Programmierern bisher zur Verfügung stand. In älteren Versionen von ActionScript gibt es keinen Ereignisablauf. Das bedeutet, dass Ereignis-Listener nur zu dem Objekt hinzugefügt werden können, mit dem ein Ereignis erzeugt wird. In ActionScript 3.0 können Sie Ereignis-Listener nicht nur dem Zielknoten, sondern auch jedem beliebigen Knoten entlang des Ereignisablaufs hinzufügen.

Die Möglichkeit, Ereignis-Listener entlang des Ereignisablaufs hinzuzufügen, ist dann hilfreich, wenn eine Benutzerschnittstellenkomponente aus mehr als einem Objekt besteht. Ein Button-Objekt enthält beispielsweise häufig ein Textobjekt mit der Schaltflächenbeschriftung. Ohne die Möglichkeit, dem Ereignisablauf Listener hinzuzufügen, müssten Sie sowohl dem Button-Objekt als auch dem Text-Objekt einen Listener hinzufügen, um sicherzustellen, dass Sie Benachrichtigungen über Mausklickereignisse für die gesamte Schaltfläche erhalten. Der Ereignisablauf ermöglicht jedoch die Definition eines einzigen Ereignis-Listeners für das Button-Objekt, mit dem Mausklickereignisse für das Text-Objekt oder für die Bereiche des Button-Objekts verarbeitet werden, die nicht durch das Text-Objekt verdeckt sind.

Nicht jedes Ereignisobjekt durchläuft jedoch alle drei Phasen des Ereignisablaufs. Einige Ereignistypen, beispielsweise `enterFrame` und `init`, werden direkt an den Zielknoten gesendet und durchlaufen weder die Empfangsphase noch die Aufstiegsphase. Die Zielobjekte anderer Ereignisse sind unter Umständen nicht Bestandteil der Anzeigeliste, z. B. Ereignisse, die an eine Instanz der Socket-Klasse gesendet werden. Diese Ereignisobjekte werden ebenfalls direkt an das Zielobjekt geleitet, ohne die Empfangsphase oder die Aufstiegsphase zu durchlaufen.

Wenn Sie das Verhalten eines bestimmten Ereignistyps ermitteln möchten, finden Sie entweder entsprechende Informationen in der API-Dokumentation oder können die Eigenschaften des jeweiligen Ereignisobjekts überprüfen. Die Überprüfung der Eigenschaften eines Ereignisobjekts wird im folgenden Abschnitt beschrieben.

Ereignisobjekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ereignisobjekte dienen im neuen Ereignisverarbeitungssystem zwei Hauptzwecken. Erstens stellen Ereignisobjekte tatsächliche Ereignisse dar, indem Informationen zu bestimmten Ereignissen in einem Eigenschaftensatz gespeichert werden. Zweitens enthalten Ereignisobjekte einen Satz von Methoden, mit denen Sie Ereignisobjekte bearbeiten und das Verhalten des Ereignis verarbeitenden Systems beeinflussen können.

Um den Zugriff auf diese Eigenschaften und Methoden zu erleichtern, ist in der Flash Player-API eine Event-Klasse definiert, die als Basisklasse für alle Ereignisobjekte dient. Die Event-Klasse definiert einen grundlegenden Satz von Eigenschaft und Methoden, die alle Ereignisobjekte gemeinsam haben.

In diesem Abschnitt werden zunächst die Eigenschaften der Event-Klasse behandelt und dann die entsprechenden Methoden beschrieben. Schließlich wird erklärt, warum es Unterklassen der Event-Klasse gibt.

Eigenschaften der Event-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Event-Klasse definiert eine Reihe schreibgeschützter Eigenschaften und Konstanten, die wichtige Informationen zu einem Ereignisobjekt enthalten. Besonders wichtig sind dabei folgende Eigenschaften:

- Der Typ von Ereignisobjekten wird durch Konstanten festgelegt und in der `Event.type`-Eigenschaft gespeichert.
- Ob das Standardverhalten eines Ereignisses unterdrückt werden kann, wird durch einen booleschen Wert angegeben und in der Eigenschaft `Event.cancelable` gespeichert.
- Die übrigen Eigenschaften enthalten Informationen zum Ereignisablauf.

Ereignisobjekttypen

Jedem Ereignisobjekt ist ein Ereignistyp zugewiesen. Ereignistypen sind in der `Event.type`-Eigenschaft als Stringwerte gespeichert. Es ist hilfreich, den Typ eines Ereignisobjekts zu kennen, damit Ihr Code Objekte verschiedener Typen voneinander unterscheiden kann. Mit dem folgenden Code wird beispielsweise festgelegt, dass die Listener-Funktion `clickHandler()` auf alle Mausklick-Ereignisobjekte reagieren soll, die an `myDisplayObject` übergeben werden:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

Der Event-Klasse selbst sind etwa zwei Dutzend Ereignistypen zugeordnet. Diese werden durch Konstanten der Event-Klasse angegeben. Einige davon finden Sie im nachstehenden Auszug aus der Definition der Event-Klasse:

```
package flash.events
{
    public class Event
    {
        // class constants
        public static const ACTIVATE:String = "activate";
        public static const ADDED:String= "added";
        // remaining constants omitted for brevity
    }
}
```

Diese Konstanten stellen eine einfache Möglichkeit dar, sich auf bestimmte Ereignistypen zu beziehen. Verwenden Sie diese Konstanten anstelle der Strings, denen sie entsprechen. Schreibfehler in einem Konstantennamen im Programmcode werden vom Compiler gemeldet. Wenn Sie jedoch Strings verwenden, werden Rechtschreibfehler beim Kompilieren u. U. nicht erkannt und können zu unerwartetem Verhalten führen. Die Fehlersuche ist möglicherweise schwierig. Verwenden Sie beispielsweise beim Hinzufügen eines Ereignis-Listeners besser:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

anstelle von:

```
myDisplayObject.addEventListener("click", clickHandler);
```

Informationen zum Standardverhalten

Ihr Code kann überprüfen, ob das Standardverhalten für ein bestimmtes Ereignisobjekt verhindert werden kann, indem er auf die `cancelable`-Eigenschaft zugreift. Die `cancelable`-Eigenschaft enthält einen booleschen Wert, der angibt, ob das Standardverhalten unterdrückt werden kann oder nicht. Bei einigen Ereignissen können Sie das Standardverhalten mit der `preventDefault()`-Methode verhindern, oder „abbrechen“. Weitere Informationen finden Sie unter „Unterdrücken des Standardverhaltens für Ereignisse“ unter „Methoden der Event-Klasse“ auf Seite 142.

Informationen zum Ereignisablauf

Die übrigen Eigenschaften der Event-Klasse enthalten wichtige Informationen zu einem Ereignisobjekt und der entsprechenden Einordnung im Ereignisablauf. Diese sind in der folgenden Liste beschrieben:

- Die `bubbles`-Eigenschaft enthält Informationen über die Bereiche des Ereignisablaufs, die das Ereignisobjekt durchläuft.
- Die `eventPhase`-Eigenschaft gibt die aktuelle Phase im Ereignisablauf an.
- Die `target`-Eigenschaft speichert einen Verweis auf das Ereignisziel.
- Die `currentTarget`-Eigenschaft speichert einen Verweis auf das Objekt in der Anzeigeliste, mit dem das Ereignisobjekt derzeit verarbeitet wird.

bubbles-Eigenschaft

Es wird von der Aufstiegsphase eines Ereignisses gesprochen, wenn das entsprechende Ereignisobjekt die Aufstiegsphase des Ereignisablaufs durchläuft. Dies bedeutet, dass das Ereignisobjekt vom Zielknoten über die Vorgängerobjekte zurückgegeben wird, bis es die Bühne erreicht. Die `Event.bubbles`-Eigenschaft speichert einen booleschen Wert, der angibt, ob das Ereignis die Aufstiegsphase durchläuft. Da alle die Aufstiegsphase durchlaufenden Ereignisse auch die Empfangsphase und die Zielphase durchlaufen, durchläuft jedes entsprechende Ereignis alle drei Phasen des Ereignisablaufs. Beim Wert `true` durchläuft das Ereignisobjekt alle drei Phasen. Wenn der Wert `false` ist, durchläuft das Objekt die Aufstiegsphase nicht.

eventPhase-Eigenschaft

Sie können die Ereignisphase für jedes Ereignisobjekt ermitteln, indem Sie die entsprechende `eventPhase`-Eigenschaft auslesen. Die `eventPhase`-Eigenschaft enthält einen vorzeichenlosen Ganzzahlwert, mit dem eine der drei Phasen des Ereignisablaufs angegeben wird. In der Flash Player-API ist eine eigene EventPhase-Klasse mit drei Konstanten für diese drei vorzeichenlosen Ganzzahlwerte definiert, wie im folgenden Codeauszug dargestellt:

```
package flash.events
{
    public final class EventPhase
    {
        public static const CAPTURING_PHASE:uint = 1;
        public static const AT_TARGET:uint = 2;
        public static const BUBBLING_PHASE:uint= 3;
    }
}
```

Diese Konstanten entsprechen den drei möglichen Werten für die `eventPhase`-Eigenschaft. Sie können diese Konstanten einsetzen, damit der Programmcode besser lesbar wird. Wenn Sie beispielsweise sicherstellen möchten, dass eine Funktion mit dem Namen `myFunc()` nur aufgerufen wird, wenn sich das Ereigniszieln in der Zielphase befindet, können Sie diese Bedingung mit dem folgenden Programmcode testen:

```
if (event.eventPhase == EventPhase.AT_TARGET)
{
    myFunc();
}
```

target-Eigenschaft

Die `target`-Eigenschaft enthält einen Verweis auf das Zielobjekt des entsprechenden Ereignisses. In einigen Fällen handelt es sich um eine direkte Zuordnung, z. B. wenn ein Mikrofon aktiviert wird und das Ziel des Ereignisobjekts das Microphone-Objekt ist. Wenn das Ziel Bestandteil der Anzeigeliste ist, muss jedoch auch die Anzeigelistenhierarchie berücksichtigt werden. Wenn ein Benutzer beispielsweise einen Mausklick an einem Punkt eingibt, an dem sich Anzeigelistenobjekte überlappen, wird in Flash Player und AIR immer das Objekt als Ereigniszieln ausgewählt, das am weitesten von der Bühne entfernt ist.

Bei komplexen SWF-Dateien – besonders solchen, bei denen Schaltflächen üblicherweise mit kleineren untergeordneten Objekten versehen sind – wird die `target`-Eigenschaft häufig nicht verwendet, da sie in der Regel nicht auf die Schaltfläche, sondern auf ein untergeordnetes Objekt verweist. In diesen Fällen ist es üblich, den Ereignis-Listener der Schaltfläche hinzuzufügen und die `currentTarget`-Eigenschaft zu verwenden, da diese auf die Schaltfläche verweist, während die `target`-Eigenschaft auch auf ein untergeordnetes Objekt der Schaltfläche verweisen kann.

currentTarget-Eigenschaft

Die `currentTarget`-Eigenschaft enthält einen Verweis auf das Objekt, mit dem das Ereignisobjekt derzeit verarbeitet wird. Es mag seltsam erscheinen, nicht zu wissen, in welchem Knoten das überprüfte Ereignisobjekt derzeit verarbeitet wird. Beachten Sie jedoch, dass Sie jedem Anzeigeobjekt im Ereignisablauf des Ereignisobjekts eine Listener-Funktion hinzufügen können und dass diese an beliebiger Stelle platziert werden kann. Darüber hinaus kann dieselbe Listener-Funktion unterschiedlichen Anzeigeobjekten hinzugefügt werden. Wenn Projekte an Umfang und Komplexität zunehmen, erhöht sich auch zunehmend der Nutzen der `currentTarget`-Eigenschaft.

Methoden der Event-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Es gibt drei Kategorien von Event-Klassenmethoden:

- Dienstprogrammmethoden, die Kopien eines Ereignisobjekts erstellen oder es in einen String konvertieren
- Ereignisablaufmethoden, mit denen Sie Ereignisobjekte aus dem Ereignisablauf entfernen können
- Standardverhaltenmethoden, die das Standardverhalten überprüfen oder feststellen, ob es verhindert wurde

Dienstprogrammmethoden der Event-Klasse

Es liegen zwei Dienstprogrammmethoden für die Event-Klasse vor. Die `clone()`-Methode ermöglicht Ihnen, Kopien eines Ereignisobjekts zu erstellen. Die `toString()`-Methode ermöglicht Ihnen, eine Stringdarstellung der Eigenschaften eines Ereignisobjekts zusammen mit deren Werten zu generieren. Beide Methoden werden intern im Ereignismodell verwendet, stehen jedoch auch zur allgemeinen Verwendung für Entwickler bereit.

Erfahrene Entwickler, die Unterklassen der Event-Klasse erstellen möchten, müssen diese beiden Methoden überschreiben und eigene Versionen implementieren, damit die neue Event-Unterklasse ordnungsgemäß verwendet werden kann.

Anhalten des Ereignisablaufs

Sie können entweder die `Event.stopPropagation()`-Methode oder die `Event.stopImmediatePropagation()`-Methode aufrufen, um die weitere Verarbeitung eines Ereignisobjekts im Ereignisablauf zu unterbinden. Beide Methoden sind fast identisch. Sie unterscheiden sich lediglich darin, ob die Abarbeitung der anderen Ereignis-Listener des aktuellen Knotens zulässig ist oder nicht:

- Die `Event.stopPropagation()`-Methode verhindert, dass das Ereignisobjekt an den nächsten Knoten übergeben wird, jedoch erst, nachdem alle anderen Ereignis-Listener des aktuellen Knotens verarbeitet wurden.
- Die `Event.stopImmediatePropagation()`-Methode verhindert, dass das Ereignisobjekt an den nächsten Knoten übergeben wird. Alle anderen Ereignis-Listener des aktuellen Knotens werden nicht mehr verarbeitet.

Keine der beiden Methoden hat Auswirkungen auf das einem Ereignis zugeordnete Standardverhalten. Verwenden Sie zum Unterdrücken des Standardverhaltens die Standardverhaltenmethoden der Event-Klasse.

Abbrechen des Standardverhaltens von Ereignissen

Die beiden Methoden, die mit dem Abbrechen von Standardverhalten zu tun haben, sind die `preventDefault()`-Methode und die `isDefaultPrevented()`-Methode. Rufen Sie die `preventDefault()`-Methode auf, um das Standardverhalten abzubrechen, das einem Ereignis zugeordnet ist. Wenn Sie überprüfen möchten, ob `preventDefault()` für ein Ereignisobjekt bereits aufgerufen wurde, verwenden Sie die `isDefaultPrevented()`-Methode. Diese gibt den Wert `true` zurück, wenn die Methode bereits aufgerufen wurde, oder andernfalls den Wert `false`.

Die `preventDefault()`-Methode wird nur ausgeführt, wenn das Standardverhalten eines Ereignisses auch wirklich unterdrückt werden kann. Ob dies der Fall ist, können Sie in der API-Dokumentation zu diesem Ereignistyp nachlesen oder in ActionScript durch Überprüfen der `cancelable`-Eigenschaft des Ereignisobjekts ermitteln.

Das Abbrechen des Standardverhaltens hat keine Auswirkungen auf den Fortgang eines Ereignisobjekts im Ereignisablauf. Verwenden Sie die Ereignisablaufmethoden der Event-Klasse, um ein Ereignisobjekt aus dem Ereignisablauf zu entfernen.

Unterklassen der Event-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für viele Ereignisse ist der allgemeine Satz der Eigenschaften, die in der Event-Klasse definiert sind, ausreichend. Andere Ereignisse weisen jedoch einzigartige Merkmale auf, die mit den in der Event-Klasse verfügbaren Eigenschaften nicht erfasst werden können. Für solche Ereignisse definiert ActionScript 3.0 mehrere Unterklassen der Event-Klasse.

Jede Unterklasse fügt zusätzliche Eigenschaften und Ereignistypen hinzu, die für diese Ereigniskategorie spezifisch sind. Beispielsweise weisen Ereignisse für Mauseingaben einige einzigartige Merkmale auf, die mit den in der Event-Klasse verfügbaren Eigenschaften nicht erfasst werden können. Die MouseEvent-Klasse erweitert die Event-Klasse durch Hinzufügen von zehn Eigenschaften für Informationen wie die Position des Mausereignisses und ob während des Mausereignisses bestimmte Tasten gedrückt wurden.

Unterklassen der Event-Klasse enthalten zudem Konstanten für die einer Unterklasse zugeordneten Ereignistypen. In der MouseEvent-Klasse sind beispielsweise Konstanten für verschiedene Mausereignistypen definiert, einschließlich der Ereignistypen `click`, `doubleClick`, `mouseDown` und `mouseUp`.

Wie im Abschnitt zu Dienstprogrammmethoden der Event-Klasse unter „Ereignisobjekte“ auf Seite 139 beschrieben, müssen Sie beim Erstellen einer Event-Unterklasse die Methoden `clone()` und `toString()` außer Kraft setzen, um die für eine Unterklasse erforderliche Funktionalität bereitzustellen.

Ereignis-Listener

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ereignis-Listener, die auch als Ereignisprozeduren bezeichnet werden, sind Funktionen, die in Flash Player und AIR als Reaktion auf bestimmte Ereignisse ausgeführt werden. Ereignis-Listener werden in zwei Schritten hinzugefügt. Zunächst erstellen Sie eine Funktion oder Klassenmethode, die in Flash Player bzw. AIR als Reaktion auf das Ereignis ausgeführt wird. Diese wird gelegentlich als Listener-Funktion oder Ereignisprozedurfunktion bezeichnet. Im zweiten Schritt rufen Sie die `addEventListener()`-Methode auf, um die Listener-Funktion beim jeweiligen Ereignisziel oder einem Anzeigelistenobjekt zu registrieren, das sich im entsprechenden Ereignisablauf befindet.

Erstellen einer Listener-Funktion

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Erstellung von Listener-Funktionen ist ein Bereich, in dem das ActionScript 3.0-Ereignismodell vom DOM-Ereignismodell abweicht. Im DOM-Ereignismodell gibt es eine klare Unterscheidung zwischen einem Ereignis-Listener und einer Listener-Funktion: Ein Ereignis-Listener ist eine Instanz einer Klasse, die die EventListener-Schnittstelle implementiert, wohingegeben eine Listener-Funktion eine Methode dieser Klasse mit dem Namen `handleEvent()` ist. Im DOM-Ereignismodell registrieren Sie anstelle der eigentlichen Listener-Funktion die Klasseninstanz, die die Listener-Funktion enthält.

Im ActionScript 3.0-Ereignismodell wird nicht zwischen Ereignis-Listenern und Listener-Funktionen unterschieden. ActionScript 3.0 verfügt über keine EventListener-Schnittstelle. Listener-Funktionen können außerhalb einer Klasse oder als Teil einer Klasse definiert werden. Darüber hinaus müssen Listener-Funktionen nicht stets den Namen `handleEvent()` tragen. Sie können mit einem beliebigen gültigen Bezeichner benannt werden. In ActionScript 3.0 registrieren Sie den Namen der eigentlichen Listener-Funktion.

Außerhalb einer Klasse definierte Listener-Funktionen

Im folgenden Programmcode wird eine einfache SWF-Datei erstellt, mit der ein rotes Quadrat angezeigt wird. Eine Listener-Funktion mit dem Namen `clickHandler()`, die nicht Bestandteil einer Klasse ist, wartet auf Mausklickereignisse für das rote Quadrat.

```
package
{
    import flash.display.Sprite;

    public class ClickExample extends Sprite
    {
        public function ClickExample()
        {
            var child:ChildSprite = new ChildSprite();
            addChild(child);
        }
    }
}

import flash.display.Sprite;
import flash.events.MouseEvent;

class ChildSprite extends Sprite
{
    public function ChildSprite()
    {
        graphics.beginFill(0xFF0000);
        graphics.drawRect(0,0,100,100);
        graphics.endFill();
        addEventListener(MouseEvent.CLICK, clickHandler);
    }
}

function clickHandler(event:MouseEvent):void
{
    trace("clickHandler detected an event of type: " + event.type);
    trace("the this keyword refers to: " + this);
}
```

Wenn ein Benutzer durch Klicken auf das Quadrat mit der entstandenen SWF-Datei interagiert, wird in Flash Player bzw. AIR die folgende Trace-Ausgabe erzeugt:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Beachten Sie, dass das Ereignisobjekt als Argument an `clickHandler()` übergeben wird. Dadurch kann mit der Listener-Funktion das Ereignisobjekt überprüft werden. In diesem Beispiel wird mit der `type`-Eigenschaft des Ereignisobjekts sichergestellt, dass es sich um ein Mausklickereignis handelt.

Im Beispiel wird auch der Wert des Schlüsselworts `this` überprüft. In diesem Fall repräsentiert `this` das globale Objekt. Dies ist schlüssig, da die Funktion außerhalb benutzerdefinierter Klassen und Objekte definiert wurde.

Als Methode einer Klasse definierte Listener-Funktionen

Das folgende Beispiel ist mit dem vorangegangenen identisch, in dem ebenfalls die ClickExample-Klasse definiert wurde. Die `clickHandler()`-Funktion ist diesmal jedoch als Methode der ChildSprite-Klasse definiert:

```
package
{
    import flash.display.Sprite;

    public class ClickExample extends Sprite
    {
        public function ClickExample()
        {
            var child:ChildSprite = new ChildSprite();
            addChild(child);
        }
    }
}

import flash.display.Sprite;
import flash.events.MouseEvent;

class ChildSprite extends Sprite
{
    public function ChildSprite()
    {
        graphics.beginFill(0xFF0000);
        graphics.drawRect(0,0,100,100);
        graphics.endFill();
        addEventListener(MouseEvent.CLICK, clickHandler);
    }
    private function clickHandler(event:MouseEvent):void
    {
        trace("clickHandler detected an event of type: " + event.type);
        trace("the this keyword refers to: " + this);
    }
}
```

Wenn ein Benutzer durch Klicken auf das rote Quadrat mit der entstandenen SWF-Datei interagiert, wird in Flash Player bzw. AIR die folgende Trace-Ausgabe erzeugt:

```
clickHandler detected an event of type: click
the this keyword refers to: [object ChildSprite]
```

Beachten Sie, dass sich das Schlüsselwort `this` auf die ChildSprite-Instanz mit dem Namen `child` bezieht. Dieses Verhalten ist anders als in ActionScript 2.0. Wenn Sie in ActionScript 2.0 Komponenten verwendet haben, erinnern Sie sich möglicherweise noch daran, dass beim Übergeben einer Klassenmethode an `UIEventDispatcher.addEventListener()` der Gültigkeitsbereich dieser Methode an die Komponente gebunden war, mit der das Ereignis gesendet wurde, und nicht an die Klasse, in der die Listener-Methode definiert wurde. Anders ausgedrückt, bezieht sich bei Verwendung dieser Technik in ActionScript 2.0 das Schlüsselwort `this` nicht auf die ChildSprite-Instanz, sondern auf die Komponente, mit der das Ereignis gesendet wird.

Dies stellte für einige Programmierer ein großes Problem dar, da sie deshalb nicht auf die anderen Methoden und Eigenschaften der Klasse zugreifen konnten, in der die Listener-Methode definiert war. Zum Umgehen des Problems konnten ActionScript 2.0-Programmierer die `mx.util.Delegate`-Klasse verwenden, um den Gültigkeitsbereich der Listener-Methode zu ändern. Dies ist nun jedoch nicht mehr erforderlich, da in ActionScript 3.0 beim Aufrufen von `addEventListener()` eine gebundene Methode erstellt wird. Dies führt dazu, dass sich das Schlüsselwort `this` auf die ChildSprite-Instanz mit dem Namen `child` bezieht und Programmierer Zugriff auf die anderen Methoden und Ereignisse der ChildSprite-Klasse haben.

Zu vermeidende Ereignis-Listener

Es liegt ein drittes Verfahren vor, bei dem Sie ein generisches Objekt mit einer Eigenschaft erstellen, die auf eine dynamisch zugewiesene Listener-Funktion verweist. Dieses Verfahren wird jedoch nicht empfohlen. Es wird hier erläutert, da es in ActionScript 2.0 ein gängiges Verfahren darstellte, es sollte jedoch in ActionScript 3.0 nicht verwendet werden. Dieses Verfahren wird nicht empfohlen, weil das Schlüsselwort `this` auf das globale Objekt und nicht auf das Listener-Objekt verweist.

Das folgende Beispiel ist mit dem vorangegangenen für die ClickExample-Klasse identisch. Die Listener-Funktion ist jedoch als Bestandteil eines generischen Objekts mit dem Namen `myListenerObj` definiert:

```
package
{
    import flash.display.Sprite;

    public class ClickExample extends Sprite
    {
        public function ClickExample()
        {
            var child:ChildSprite = new ChildSprite();
            addChild(child);
        }
    }
}

import flash.display.Sprite;
import flash.events.MouseEvent;

class ChildSprite extends Sprite
{
    public function ChildSprite()
    {
        graphics.beginFill(0xFF0000);
        graphics.drawRect(0,0,100,100);
        graphics.endFill();
        addEventListener(MouseEvent.CLICK, myListenerObj.clickHandler);
    }
}

var myListenerObj:Object = new Object();
myListenerObj.clickHandler = function (event:MouseEvent):void
{
        trace("clickHandler detected an event of type: " + event.type);
        trace("the this keyword refers to: " + this);
}
```

Das Ergebnis der Trace-Ausgabe ist wie folgt:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Es wäre zu erwarten, dass sich `this` auf `myListenerObj` bezieht und `[object Object]` ausgegeben wird. Stattdessen wird auf das globale Objekt verwiesen. Wenn Sie den Namen einer dynamischen Eigenschaft als Argument an `addEventListener()` übergeben, kann in Flash Player oder AIR keine gebundene Methode erstellt werden. Der Grund hierfür ist, dass Sie beim Übergeben des Parameters `listener` lediglich die Speicheradresse der Listener-Funktion übergeben. In Flash Player und AIR kann diese Speicheradresse nicht mit der `myListenerObj`-Instanz verbunden werden .

Verwalten von Ereignis-Listenern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können Listener-Funktionen mithilfe der Methoden der IEventDispatcher-Schnittstelle verwalten. Die IEventDispatcher-Schnittstelle ist die ActionScript 3.0-Variante der EventTarget-Schnittstelle des DOM-Ereignismodells. Obwohl der Name „IEventDispatcher" darauf hinzuweisen scheint, dass der Hauptzweck dieser Schnittstelle das Senden von Ereignisobjekten ist, werden die Methoden dieser Klasse in Wahrheit wesentlich häufiger zum Registrieren, Überprüfen und Entfernen von Ereignis-Listenern eingesetzt. Die IEventDispatcher-Schnittstelle definiert fünf Methoden, die im folgenden Codebeispiel dargestellt sind:

```
package flash.events
{
    public interface IEventDispatcher
    {
        function addEventListener(eventName:String,
                        listener:Object,
                        useCapture:Boolean=false,
                        priority:Integer=0,
                        useWeakReference:Boolean=false):Boolean;

        function removeEventListener(eventName:String,
                    listener:Object,
                    useCapture:Boolean=false):Boolean;

        function dispatchEvent(eventObject:Event):Boolean;

        function hasEventListener(eventName:String):Boolean;
        function willTrigger(eventName:String):Boolean;
    }
}
```

Die Flash Player-API implementiert die IEventDispatcher-Schnittstelle mithilfe der EventDispatcher-Klasse. Diese dient als Basisklasse für alle Klassen, die ein Ereignisziel oder Teil des Ereignisablaufs sein können. Beispielsweise erbt die DisplayObject-Klasse von der EventDispatcher-Klasse. Das bedeutet, dass alle Objekte in der Anzeigeliste Zugriff auf die Methoden der IEventDispatcher-Schnittstelle haben.

Hinzufügen von Ereignis-Listenern

Die `addEventListener()`-Methode ist die wichtigste Methode der IEventDispatcher-Schnittstelle. Sie setzen sie ein, um Listener-Funktionen zu registrieren. Die beiden erforderlichen Parameter sind `type` und `listener`. Verwenden Sie den Parameter `type`, um den Ereignistyp anzugeben. Mit dem Parameter `listener` geben Sie die Listener-Funktion an, die beim Auftreten eines Ereignisses ausgeführt wird. Der Parameter `listener` kann ein Verweis auf eine Funktion oder eine Klassenmethode sein.

Verwenden Sie keine Klammer, wenn Sie den `listener`-Parameter spezifizieren. Beim folgenden Aufruf der `addEventListener()`-Methode wird die `clickHandler()`-Funktion beispielsweise ohne Klammern angegeben:

```
addEventListener(MouseEvent.CLICK, clickHandler)
```

Mit dem Parameter `useCapture` der `addEventListener()`-Methode können Sie die Ereignisablaufphase festlegen, in der der Listener aktiv ist. Wenn `useCapture` den Wert `true` hat, ist der Listener während der Empfangsphase des Ereignisablaufs aktiv. Wenn `useCapture` den Wert `false` hat, ist der Listener während der Zielphase und der Aufstiegsphase des Ereignisablaufs aktiv. Damit in allen Phasen des Ereignisablaufs auf das Ereignis gewartet wird, müssen Sie `addEventListener()` zweimal aufrufen: einmal mit `useCapture` für den Wert`true` und einmal mit`useCapture` für den Wert `false`.

Der Parameter `priority` der `addEventListener()`-Methode ist kein offizieller Bestandteil des DOM3-Ereignismodells. Er wurde in ActionScript 3.0 aufgenommen, um eine höhere Flexibilität beim Organisieren von Ereignis-Listenern zu ermöglichen. Beim Aufrufen von `addEventListener()` können Sie die Priorität des jeweiligen Ereignis-Listeners festlegen, indem Sie als `priority`-Parameter einen Ganzzahlwert übergeben. Der Standardwert ist 0. Sie können jedoch sowohl negative als auch positive Werte angeben. Je größer die Zahl ist, desto eher wird der entsprechende Ereignis-Listener abgearbeitet. Ereignis-Listener mit derselben Priorität werden in der Reihenfolge des Hinzufügens abgearbeitet. Je eher ein Listener hinzugefügt wurde, desto eher wird er auch abgearbeitet.

Mit dem Parameter `useWeakReference` können Sie festlegen, ob es sich um einen schwachen oder normalen Verweis auf die Listener-Funktion handelt. Durch Festlegen von `true` für diesen Parameter können Sie Situationen vermeiden, in denen Listener-Funktionen im Speicher zurückbleiben, auch wenn sie nicht mehr benötigt werden. In Flash Player und AIR wird ein Verfahren mit der Bezeichnung *Garbage Collection* (automatische Speicherbereinigung) verwendet, um nicht mehr verwendete Objekte aus dem Speicher zu entfernen. Objekte gelten als nicht mehr verwendet, wenn keine Verweise auf sie vorhanden sind. Bei der Speicherbereinigung werden schwache Verweise nicht berücksichtigt. Das bedeutet, dass Listener-Funktionen mit ausschließlich schwachen Verweisen bei der Speicherbereinigung entfernt werden.

Entfernen von Ereignisprozeduren

Mit der `removeEventListener()`-Methode können Sie einen Ereignis-Listener entfernen, den Sie nicht mehr benötigen. Es empfiehlt sich, alle nicht mehr benötigten Listener zu entfernen. Erforderliche Parameter sind `eventName` und `listener`, die identisch sind mit den erforderlichen Parametern für die `addEventListener()`-Methode. Rufen Sie sich in Erinnerung, dass in allen Phasen des Ereignisablaufs auf das Ereignis gewartet werden kann, wenn Sie `addEventListener()` zweimal aufrufen: einmal mit dem Wert `true` für `useCapture` und einmal mit dem Wert `false`. Damit beide Ereignis-Listener wieder entfernt werden, müssen Sie `removeEventListener()` ebenfalls zweimal aufrufen: einmal mit dem Wert `true` für `useCapture` und einmal mit dem Wert `false`.

Auslösen von Ereignissen

Erfahrene Programmierer können die `dispatchEvent()`-Methode einsetzen, um ein benutzerdefiniertes Ereignisobjekt in den Ereignisablauf einzufügen. Der einzige für diese Methode zulässige Parameter ist ein Verweis auf ein Ereignisobjekt, das eine Instanz der Event-Klasse oder einer ihrer Unterklassen sein muss. Nach dem Auslösen des Ereignisses ist als `target`-Eigenschaft des Ereignisobjekts das Objekt eingetragen, für das `dispatchEvent()` aufgerufen wurde.

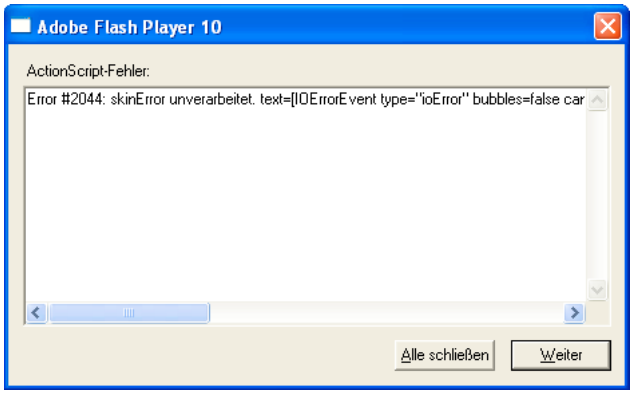
Überprüfen auf vorhandene Ereignis-Listener

Die letzten beiden Methoden der IEventDispatcher-Schnittstelle stellen nützliche Informationen über das Vorhandensein von Ereignis-Listenern bereit. Die `hasEventListener()`-Methode gibt `true` zurück, wenn für einen bestimmten Ereignistyp und das angegebene Anzeigelistenobjekt ein Ereignis-Listener gefunden wurde. Die `willTrigger()`-Methode gibt ebenfalls `true` zurück, wenn für ein bestimmtes Anzeigelistenobjekt ein Ereignis-Listener gefunden wurde. Bei `willTrigger()` wird jedoch nicht nur dieses eine Objekt in der Anzeigeliste überprüft, sondern für alle Phasen des Ereignisablaufs auch alle Vorgänger dieses Objekts.

Fehlerereignisse ohne Listener

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der primäre Mechanismus für die Fehlerverarbeitung in ActionScript 3.0 sind nicht Ereignisse, sondern Ausnahmen. Die Ausnahmenverarbeitung funktioniert jedoch nicht bei asynchronen Vorgängen, z. B. beim Laden von Dateien. Wenn während eines asynchronen Vorgangs ein Fehler auftritt, wird in Flash Player und AIR ein Fehlerereignisobjekt ausgelöst. Wenn Sie keinen Listener für dieses Fehlerereignis erstellt haben, wird in der Debugger-Versionen von Flash Player und AIR ein Dialogfeld mit Informationen zum Fehler angezeigt. In der Debugger-Version von Flash Player wird beispielsweise das folgende Dialogfeld mit einer Fehlerbeschreibung angezeigt, wenn die Anwendung versucht, eine Datei von einer ungültigen URL zu laden:



Die meisten Fehlerereignisse basieren auf der ErrorEvent-Klasse und verfügen daher über eine `text`-Eigenschaft, die zum Speichern der in Flash Player oder AIR angezeigten Fehlermeldung verwendet wird. Die beiden Ausnahmen sind die StatusEvent- und die NetStatusEvent-Klasse. Beide Klassen verfügen über eine `level`-Eigenschaft (`StatusEvent.level` und `NetStatusEvent.info.level`). Wenn diese `level`-Eigenschaft den Wert `error` aufweist, wird bei diesen Ereignistypen von Fehlerereignissen ausgegangen.

Fehlerereignisse führen nicht zum Abbruch der Ausführung einer SWF-Datei. Sie manifestieren sich nur als Dialogfelder in den Debugger-Versionen des Browser-Plug-Ins und des eigenständigen Players, als Meldung im Bedienfeld „Ausgabe“ des Authoring-Players und als Eintrag in der Protokolldatei für Adobe Flash Builder. In den normalen Versionen von Flash Player bzw. AIR werden Fehlerereignisse in keiner Form angezeigt.

Beispiel für die Ereignisverarbeitung: Alarm Clock

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Beispielanwendung „Alarm Clock“ besteht aus einer Uhr, für die Benutzer eine Weckzeit sowie eine Meldung angeben können, die zur Weckzeit angezeigt werden soll. Die Beispielanwendung „Alarm Clock“ basiert auf der Anwendung „SimpleClock“ aus „Arbeiten mit Datum und Zeit“ auf Seite 1. Dieses Beispiel veranschaulicht mehrere Aspekte der Verwendung von Ereignissen in ActionScript 3.0:

- Warten und Reagieren auf Ereignisse
- Benachrichtigen von Listenern über Ereignisse
- Erstellen benutzerdefinierter Ereignistypen

Die Flash Professional-Anwendungsdateien für dieses Beispiel finden Sie unter http://www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Flex-Anwendungsdateien für dieses Beispiel finden Sie unter http://www.adobe.com/go/as3examples_de. Die Dateien der Anwendung „Alarm Clock“ befinden sich im Ordner „Samples/AlarmClock“. Zur Anwendung gehören folgende Dateien:

Datei	Beschreibung
AlarmClockApp.mxml oder AlarmClockApp.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/clock/AlarmClock.as	Eine Klasse, die die SimpleClock-Klasse um Weckfunktionen erweitert.
com/example/programmingas3/clock/AlarmEvent.as	Eine benutzerdefinierte Event-Klasse (eine Unterklasse von flash.events.Event), die als Ereignisobjekt für das `alarm`-Ereignis der AlarmClock-Klasse dient.
com/example/programmingas3/clock/AnalogClockFace.as	Zeichnet ein rundes Zifferblatt sowie Stunden-, Minuten- und Sekundenzeiger anhand der aktuellen Zeit (siehe Beschreibung im SimpleClock-Beispiel).
com/example/programmingas3/clock/SimpleClock.as	Eine Uhrenschnittstellenkomponente mit einfacher Zeitmessungsfunktionalität (siehe Beschreibung im SimpleClock-Beispiel).

Überblick über die Beispielanwendung „Alarm Clock“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für die primäre Funktionalität der Uhr in diesem Beispiel (einschließlich der Zeitmessung und der Zifferblattanzeige) wird die unter „Beispiel für Datum und Uhrzeit: Einfache Analoguhr“ auf Seite 6 beschriebene Anwendung „SimpleClock“ verwendet. Die AlarmClock-Klasse erweitert die SimpleClock-Klasse aus diesem Beispiel um die für einen Wecker benötigten Funktionen. Dazu gehören das Festlegen einer Weckzeit und das Ausgeben einer Benachrichtigung, wenn der Wecker klingelt.

Ereignisse werden zur Bereitstellung von Benachrichtigungen beim Eintreten bestimmter Bedingungen eingesetzt. Die AlarmClock-Klasse stellt das Alarm-Ereignis bereit, auf das andere Objekte warten können, um dann die gewünschten Aktionen auszuführen. Zusätzlich wird mit der AlarmClock-Klasse eine Instanz der Timer-Klasse verwendet, um zu ermitteln, zu welchem Zeitpunkt der Weckalarm ausgelöst werden muss. Wie die AlarmClock-Klasse stellt auch die Timer-Klasse ein Ereignis bereit, um andere Objekte (in diesem Fall eine AlarmClock-Instanz) zu benachrichtigen, wenn eine bestimmte Zeit vergangen ist. Wie bei den meisten ActionScript-Anwendungen bilden Ereignisse einen wichtigen Teil der Funktionalität der Beispielanwendung „Alarm Clock“.

Auslösen des Weckalarms

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie bereits erwähnt, bezieht sich die einzige tatsächlich mit der AlarmClock-Klasse bereitgestellte Funktionalität auf das Festlegen der Weckzeit und das Auslösen des Weckalarms. Die integrierte Timer-Klasse (flash.utils.Timer) ermöglicht es Entwicklern, Programmcode festzulegen, der nach Ablauf einer bestimmten Zeit ausgeführt wird. Die AlarmClock-Klasse verwendet eine Timer-Instanz, um zu ermitteln, zu welchem Zeitpunkt der Weckalarm ausgelöst werden muss.

```
import flash.events.TimerEvent;
import flash.utils.Timer;

/**
 * The Timer that will be used for the alarm.
 */
public var alarmTimer:Timer;
...
/**
 * Instantiates a new AlarmClock of a given size.
 */
public override function initClock(faceSize:Number = 200):void
{
    super.initClock(faceSize);
    alarmTimer = new Timer(0, 1);
    alarmTimer.addEventListener(TimerEvent.TIMER, onAlarm);
}
```

Die in der AlarmClock-Klasse definierte Timer-Instanz trägt die Bezeichnung `alarmTimer`. Mit der `initClock()`-Methode, durch die die erforderlichen Initialisierungsvorgänge der AlarmClock-Instanz vorgenommen werden, wird die Variable `alarmTimer` in zwei Schritten vorbereitet. Zunächst wird die Variable mit Parametern instanziiert, durch die in der Timer-Instanz festgelegt wird, dass 0 Millisekunden gewartet und das Timer-Ereignis nur einmal ausgelöst wird. Nach dem Instanziieren von `alarmTimer` wird die `addEventListener()`-Methode dieser Variablen aufgerufen, um anzugeben, dass auf das `timer`-Ereignis dieser Variablen gewartet werden soll. Timer-Instanzen senden das entsprechende `timer`-Ereignis, nachdem eine festgelegte Zeit vergangen ist. Die AlarmClock-Klasse benötigt eine Benachrichtigung, zu welchem Zeitpunkt das `timer`-Ereignis ausgelöst wird, um den Weckalarm auszulösen. Durch Aufrufen von `addEventListener()` wird AlarmClock als Listener bei `alarmTimer` registriert. Die beiden Parameter geben an, dass die AlarmClock-Klasse auf das `timer`-Ereignis warten soll (mithilfe der Konstanten `TimerEvent.TIMER`) und dass beim Eintreten des Ereignisses als Reaktion die `onAlarm()`-Methode der AlarmClock-Klasse aufgerufen werden soll.

Um den Weckalarm festzulegen, wird die `setAlarm()`-Methode der AlarmClock-Klasse wie folgt aufgerufen:

```
/**
 * Sets the time at which the alarm should go off.
 * @param hour The hour portion of the alarm time.
 * @param minutes The minutes portion of the alarm time.
 * @param message The message to display when the alarm goes off.
 * @return The time at which the alarm will go off.
 */
public function setAlarm(hour:Number = 0, minutes:Number = 0, message:String = "Alarm!"):Date
{
    this.alarmMessage = message;
    var now:Date = new Date();
    // Create this time on today's date.
    alarmTime = new Date(now.fullYear, now.month, now.date, hour, minutes);

    // Determine if the specified time has already passed today.
    if (alarmTime <= now)
    {
        alarmTime.setTime(alarmTime.time + MILLISECONDS_PER_DAY);
    }

    // Stop the alarm timer if it's currently set.
    alarmTimer.reset();
    // Calculate how many milliseconds should pass before the alarm should
    // go off (the difference between the alarm time and now) and set that
    // value as the delay for the alarm timer.
    alarmTimer.delay = Math.max(1000, alarmTime.time - now.time);
    alarmTimer.start();

    return alarmTime;
}
```

Mit dieser Methode werden mehrere Vorgänge durchgeführt, darunter Speichern der Meldung für den Weckalarm und Erstellen eines Date-Objekts (`alarmTime`), das den genauen Zeitpunkt angibt, an dem der Weckalarm ausgelöst werden soll. Von größter Bedeutung sind die Festlegung und Aktivierung des Timers der Variablen `alarmTimer`, die in den letzten Programmzeilen der Methode erfolgen. Zuerst wird die `reset()`-Methode der Variablen aufgerufen, um den Timer anzuhalten und zurückzusetzen, falls er bereits aktiviert ist. Anschließend wird die aktuelle Uhrzeit (angegeben durch die Variable `now`) vom Wert der Variablen `alarmTime` subtrahiert, um zu ermitteln, wie viele Millisekunden vergehen sollen, bevor der Weckalarm ausgelöst wird. Das `timer`-Ereignis der Timer-Klasse wird nicht zu einem absoluten Zeitpunkt ausgelöst. Deshalb wird der `delay`-Eigenschaft von `alarmTimer` diese relative Zeitdifferenz zugewiesen. Schließlich wird die `start()`-Methode aufgerufen, um den Timer zu starten.

Nachdem die angegebene Zeit vergangen ist, löst `alarmTimer` das `timer`-Ereignis aus. Da die `onAlarm()`-Methode der AlarmClock-Klasse als Listener für dieses Ereignis registriert ist, wird beim Eintreten des `timer`-Ereignisses `onAlarm()` aufgerufen.

```
/**
 * Called when the timer event is dispatched.
 */
public function onAlarm(event:TimerEvent):void
{
    trace("Alarm!");
    var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
    this.dispatchEvent(alarm);
}
```

Eine als Ereignis-Listener registrierte Methode muss mit der richtigen Signatur (Anzahl und Typ der Parameter sowie Rückgabetyp der Methode) definiert werden. Als Listener für das `timer`-Ereignis der Timer-Klasse muss für eine Methode ein Parameter des Datentyps „TimerEvent“ (flash.events.TimerEvent) definiert sein, einer Unterklasse der Event-Klasse. Wenn die Timer-Instanz die zugehörigen Ereignis-Listener aufruft, wird als Ereignisobjekt eine TimerEvent-Instanz übergeben.

Weckalarmbenachrichtigungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie die Timer-Klasse stellt die AlarmClock-Klasse ein Ereignis bereit, mit dessen Hilfe Benachrichtigungen über das Auslösen des Weckalarms empfangen werden können. Damit eine Klasse das Framework zur Ereignisverarbeitung von ActionScript verwenden kann, muss sie die flash.events.IEventDispatcher-Schnittstelle implementieren. Dies wird am häufigsten durch Erweitern der flash.events.EventDispatcher-Klasse (oder einer ihrer Unterklassen) erreicht. Diese stellt eine Standardimplementierung von IEventDispatcher bereit. Wie bereits beschrieben, erweitert die AlarmClock-Klasse die SimpleClock-Klasse, die ihrerseits (über eine Vererbungskette) die EventDispatcher-Klasse erweitert. Das bedeutet, dass in der AlarmClock-Klasse bereits die Funktionalität integriert ist, die zum Bereitstellen von Ereignissen benötigt wird.

Anderer Programmcode kann für Benachrichtigungen über das `alarm`-Ereignis der AlarmClock-Klasse durch Aufrufen der `addEventListener()`-Methode registriert werden, die AlarmClock von EventDispatcher erbt. Wenn eine AlarmClock-Instanz anderen Programmcode über das Auslösen des `alarm`-Ereignisses benachrichtigt, erfolgt dies durch Aufrufen der `dispatchEvent()`-Methode, die ebenfalls von EventDispatcher übernommen wurde.

```
var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
this.dispatchEvent(alarm);
```

Diese Codezeilen stammen aus der `onAlarm()`-Methode der AlarmClock-Klasse (die bereits an anderer Stelle vollständig aufgeführt wurde). Die `dispatchEvent()`-Methode der AlarmClock-Instanz wird aufgerufen, die wiederum alle registrierten Listener darüber benachrichtigt, dass das `alarm`-Ereignis der AlarmClock-Instanz ausgelöst wurde. Bei dem an `dispatchEvent()` übergebenen Parameter handelt es sich um das Ereignisobjekt, das an die Listener-Methoden weitergeleitet wird. Im vorliegenden Fall ist es eine Instanz der AlarmEvent-Klasse, einer Event-Unterklasse, die speziell für dieses Beispiel erstellt wurde.

Bereitstellen benutzerdefinierter Alarmereignisse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Alle Ereignis-Listener empfangen einen Ereignisobjektparameter mit Informationen über das jeweils ausgelöste Ereignis. In vielen Fällen ist das Ereignisobjekt eine Instanz der Event-Klasse. Gelegentlich ist es jedoch hilfreich, Ereignis-Listenern zusätzliche Informationen zur Verfügung zu stellen. Dies wird üblicherweise durch die Definition einer neuen Klasse (einer Unterklasse der Event-Klasse) erreicht, deren Instanz dann als Ereignisobjekt verwendet wird. In diesem Beispiel wird als Ereignisobjekt eine AlarmEvent-Instanz verwendet, wenn das `alarm`-Ereignis der AlarmClock-Klasse ausgelöst wird. Die hier dargestellte AlarmEvent-Klasse stellt zusätzliche Informationen über das `alarm`-Ereignis bereit, d. h. die Meldung für den Weckalarm:

```
import flash.events.Event;

/**
 * This custom Event class adds a message property to a basic Event.
 */
public class AlarmEvent extends Event
{
    /**
     * The name of the new AlarmEvent type.
     */
    public static const ALARM:String = "alarm";

    /**
     * A text message that can be passed to an event handler
     * with this event object.
     */
    public var message:String;

    /**
     *Constructor.
     *@param message The text to display when the alarm goes off.
     */
    public function AlarmEvent(message:String = "ALARM!")
    {
        super(ALARM);
        this.message = message;
    }
    ...
}
```

Die beste Möglichkeit zum Erstellen einer benutzerdefinierten Ereignisobjektklasse ist das Definieren einer Klasse, die die Event-Klasse erweitert, wie dies im vorangegangenen Beispiel dargestellt ist. Zur Ergänzung der geerbten Funktionalität definiert die AlarmEvent-Klasse die `message`-Eigenschaft, die den Text der dem Ereignis zugeordneten Meldung für den Weckalarm enthält. Der Wert von `message` wird als Parameter an den AlarmEvent-Konstruktor übergeben. Die AlarmEvent-Klasse definiert auch die Konstante `ALARM`, mit der beim Aufrufen der `addEventListener()`-Methode der AlarmClock-Klasse auf dieses bestimmte Ereignis (`alarm`) Bezug genommen werden kann.

Zusätzlich zum Hinzufügen benutzerdefinierter Funktionalität muss jede Unterklasse der Event-Klasse die als Bestandteil des ActionScript-Frameworks zur Ereignisverarbeitung geerbte `clone()`-Methode überschreiben. Event-Unterklassen können auch optional die übernommene `toString()`-Methode überschreiben, um in den Rückgabewert der `toString()`-Methode auch die benutzerdefinierten Ereigniseigenschaften aufzunehmen.

```
    /**
     * Creates and returns a copy of the current instance.
     * @return A copy of the current instance.
     */
    public override function clone():Event
    {
        return new AlarmEvent(message);
    }

    /**
     * Returns a String containing all the properties of the current
     * instance.
     * @return A string representation of the current instance.
     */
    public override function toString():String
    {
        return formatToString("AlarmEvent", "type", "bubbles", "cancelable", "eventPhase",
"message");
    }
```

Die überschriebene `clone()`-Methode muss eine neue Instanz der benutzerdefinierten Event-Unterklasse zurückgeben, bei der alle benutzerdefinierten Eigenschaften mit der aktuellen Instanz übereinstimmen. In der überschriebenen `toString()`-Methode kommt die Dienstprogrammmethode `formatToString()` (von Event geerbt) zum Einsatz, um einen String mit dem Namen des benutzerdefinierten Typs sowie den Namen und Werten aller Eigenschaften bereitzustellen.

Kapitel 9: Verwenden von Anwendungsdomänen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der ApplicationDomain-Klasse kann eine Tabelle mit ActionScript 3.0-Definitionen gespeichert werden. Der gesamte Code in einer SWF-Datei ist so definiert, dass er sich in einer Anwendungsdomäne befindet. Mit Anwendungsdomänen werden Klassen unterteilt, die sich in der gleichen Sicherheitsdomäne befinden. Dies ermöglicht mehrere Definitionen der gleichen Klasse und die Wiederverwendung übergeordneter Definitionen in untergeordneten Klassen.

Sie können Anwendungsdomänen beim Laden einer externen SWF-Datei, die in ActionScript 3.0 programmiert ist, über API-Funktionen der Loader-Klasse laden. (Beachten Sie, dass Anwendungsdomänen beim Laden von Bildern oder in ActionScript 1.0 oder ActionScript 2.0 programmierten SWF-Dateien nicht verwendet werden können.) Alle ActionScript 3.0-Definitionen in der geladenen Klasse werden in der Anwendungsdomäne gespeichert. Beim Laden einer SWF-Datei können Sie angeben, dass die Datei in der gleichen Anwendungsdomäne wie das Loader-Objekt gespeichert wird. Setzen Sie dazu den `applicationDomain`-Parameter des LoaderContext-Objekts auf `ApplicationDomain.currentDomain`. Durch Einfügen der geladenen SWF-Datei in die gleiche Anwendungsdomäne können Sie direkt auf die zugehörigen Klassen zugreifen. Dies kann nützlich sein, wenn die geladene SWF-Datei eingebettete Medien enthält, auf die Sie über die zugeordneten Klassennamen zugreifen können, oder wenn Sie auf die Methoden der geladenen SWF-Datei zugreifen möchten.

Das folgende Beispiel geht davon aus, dass Zugriff auf eine separate Greeter.swf-Datei besteht, welche eine öffentliche Methode mit dem Namen welcome() definiert:

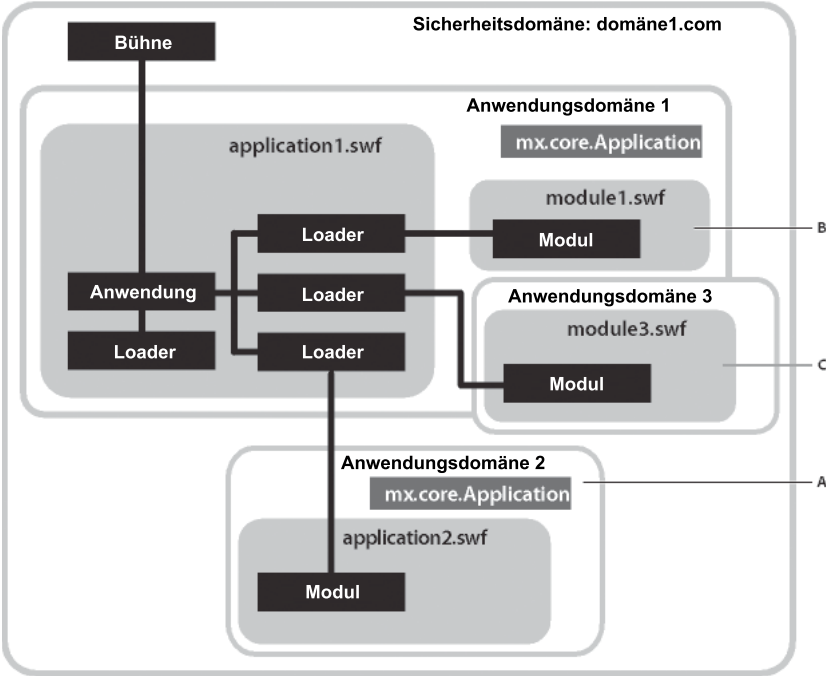
```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLRequest;
    import flash.system.ApplicationDomain;
    import flash.system.LoaderContext;

    public class ApplicationDomainExample extends Sprite
    {
        private var ldr:Loader;
        public function ApplicationDomainExample()
        {
            ldr = new Loader();
            var req:URLRequest = new URLRequest("Greeter.swf");
            var ldrContext:LoaderContext = new LoaderContext(false,
ApplicationDomain.currentDomain);
            ldr.contentLoaderInfo.addEventListener(Event.COMPLETE, completeHandler);
            ldr.load(req, ldrContext);
        }
        private function completeHandler(event:Event):void
        {
            var myGreeter:Class = ApplicationDomain.currentDomain.getDefinition("Greeter") as
Class;
            var myGreeter:Greeter = Greeter(event.target.content);
            var message:String = myGreeter.welcome("Tommy");
            trace(message); // Hello, Tommy
        }
    }
}
```

Weitere Informationen finden Sie auch im Beispiel zur ApplicationDomain-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Beachten Sie beim Verwenden von Anwendungsdomänen die folgenden Punkte:

- Der gesamte Code in einer SWF-Datei ist so definiert, dass er sich in einer Anwendungsdomäne befindet. Die *aktuelle Domäne* ist die Domäne, in der die Hauptanwendung ausgeführt wird. Die *Systemdomäne* enthält alle Anwendungsdomänen, einschließlich der aktuellen Domäne, d. h. alle Klassen von Flash Player.
- Mit Ausnahme der Systemdomäne sind alle Anwendungsdomänen mit einer übergeordneten Domäne verknüpft. Die Systemdomäne ist die übergeordnete Domäne der Anwendungsdomäne der Hauptanwendung. Geladene Klassen werden nur definiert, wenn sie in der jeweils übergeordneten Klasse nicht bereits definiert sind. Die Definition einer geladenen Klasse kann nicht mit einer neueren Definition überschrieben werden.

In der folgenden Abbildung ist eine Anwendung dargestellt, in der Inhalte aus verschiedenen SWF-Dateien in einer einzelnen Domäne, „domain1.com“, geladen werden. Je nach geladenen Inhalten können unterschiedliche Anwendungsdomänen verwendet werden. Im folgenden Text wird der Code beschrieben, der zum Festlegen der entsprechenden Anwendungsdomäne für jede SWF-Datei in der Anwendung verwendet wird.



A. *Verwendung A* ***B.*** *Verwendung B* ***C.*** *Verwendung C*

Die Datei „application1.swf" ist die Datei der Hauptanwendung. Sie enthält Loader-Objekte, über die Inhalte aus anderen SWF-Dateien geladen werden. In diesem Szenario ist die Anwendungsdomäne 1 die aktuelle Domäne. Verwendung A, Verwendung B und Verwendung C sind verschiedene Verfahren zum Festlegen der geeigneten Anwendungsdomäne für alle SWF-Dateien einer Anwendung.

Verwendung A Unterteilung der untergeordneten SWF-Datei durch Erstellen einer untergeordneten Domäne der Systemdomäne. In der Abbildung wird Anwendungsdomäne 2 als untergeordnete Domäne der Systemdomäne erstellt. Die Datei „application2.swf" wird in Anwendungsdomäne 2 geladen, die entsprechenden Klassendefinitionen sind daher von den in der Datei „application1.swf" definierten Klassen getrennt.

Eine Verwendungsmöglichkeit dieses Verfahrens besteht darin, dass in einer älteren Anwendung dynamisch eine neuere Version der gleichen Anwendung ohne Konflikte geladen werden kann. Obwohl die gleichen Klassennamen verwendet werden, treten keine Konflikte auf, da die Klassen in verschiedene Anwendungsdomänen unterteilt werden.

Mit dem folgenden Code wird eine Anwendungsdomäne erstellt, die der Systemdomäne untergeordnet ist, und das Laden einer SWF-Datei gestartet, die diese Anwendungsdomäne verwendet:

```
var appDomainA:ApplicationDomain = new ApplicationDomain();

var contextA:LoaderContext = new LoaderContext(false, appDomainA);
var loaderA:Loader = new Loader();
loaderA.load(new URLRequest("application2.swf"), contextA);
```

Verwendung B: Hinzufügen neuer Klassendefinitionen zu aktuellen Klassendefinitionen. Als Anwendungsdomäne von „module1.swf" ist die aktuelle Domäne (Anwendungsdomäne 1) festgelegt. Dadurch können Sie neue Klassendefinitionen zu den aktuellen Klassendefinitionen der Anwendung hinzufügen. Dies kann bei einer gemeinsam genutzten Laufzeitbibliothek der Hauptanwendung eingesetzt werden. Die geladene SWF-Datei wird als gemeinsam genutzte Remote-Bibliothek (RSL, Remote Shared Library) behandelt. Mit diesem Verfahren können Sie gemeinsam genutzte Remote-Bibliotheken mit einem Preloader laden, bevor die Anwendung startet.

Mit dem folgenden Code wird eine SWF-Datei geladen, wobei die Anwendungsdomäne auf die aktuelle Domäne festgelegt wird:

```
var appDomainB:ApplicationDomain = ApplicationDomain.currentDomain;

var contextB:LoaderContext = new LoaderContext(false, appDomainB);
var loaderB:Loader = new Loader();
loaderB.load(new URLRequest("module1.swf"), contextB);
```

Verwendung C: Erstellen einer neuen untergeordneten Domäne der aktuellen Domäne mithilfe der Klassendefinitionen der übergeordneten Domäne. Die Anwendungsdomäne von „module3.swf“ ist eine untergeordnete Domäne der aktuellen Domäne. In der untergeordneten Domäne werden alle Klassen der übergeordneten Domäne verwendet. Dieses Verfahren kann beispielsweise für ein Modul einer Rich Internet Application (RIA) für mehrere Bildschirme verwendet werden, die als untergeordnete Domäne der Hauptanwendung geladen wird und bei der die Typen der Hauptanwendung verwendet werden. Wenn Sie sicherstellen, dass alle Klassen immer so aktualisiert werden, dass sie abwärts kompatibel sind, und wenn die Anwendung immer aktueller ist als die geladenen Objekte, werden in den untergeordneten Domänen die übergeordneten Klassenversionen verwendet. Durch die Verwendung einer neuen Anwendungsdomäne können darüber hinaus alle Klassendefinitionen bei der automatischen Speicherbereinigung (Garbage Collection) entfernt werden, wenn Sie sicherstellen, dass keine Verweise auf die untergeordnete SWF-Datei vorhanden sind.

Bei diesem Verfahren können geladene Module die Singleton-Objekte des Loaders und die statischen Klassenmitglieder gemeinsam verwenden.

Mit dem folgenden Code wird eine neue untergeordnete Domäne der aktuellen Domäne erstellt und eine SWF-Datei geladen, die diese Anwendungsdomäne verwendet:

```
var appDomainC:ApplicationDomain = new ApplicationDomain(ApplicationDomain.currentDomain);

var contextC:LoaderContext = new LoaderContext(false, appDomainC);
var loaderC:Loader = new Loader();
loaderC.load(new URLRequest("module3.swf"), contextC);
```

Kapitel 10: Programmieren von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

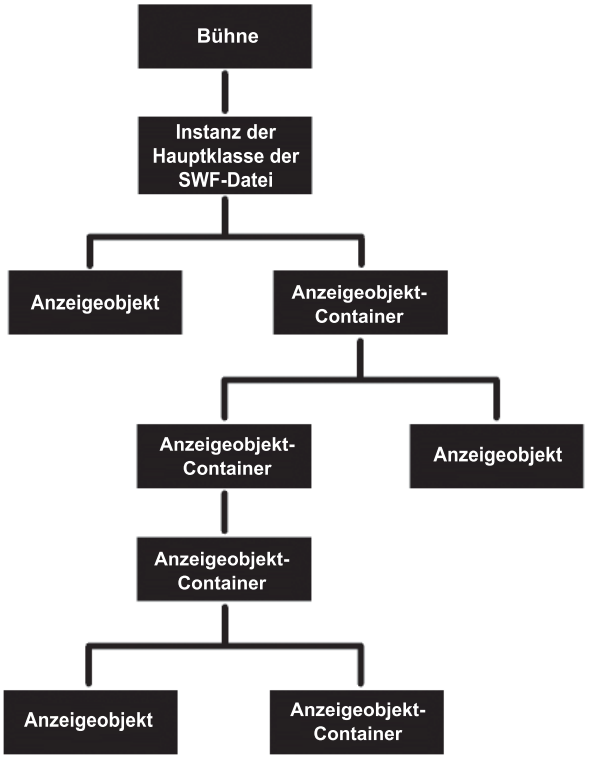
Visuelle Elemente werden in Adobe® ActionScript® 3.0 programmiert, indem Sie mit Anzeigeobjekten auf der Bühne arbeiten. Zum Beispiel können Sie mit der ActionScript-API für die Anzeigeprogrammierung Anzeigeobjekte verschieben, entfernen und anordnen, Filter und Masken anwenden, Vektor- und Bitmapgrafiken zeichnen und dreidimensionale Transformationen ausführen. Die Primärklassen für die Anzeigeprogrammierung gehören zum flash.display-Paket.

Hinweis: *In Adobe® AIR™ steht das HTMLoader-Objekt für das Rendern und Anzeigen von HTML-Inhalt zur Verfügung. Das HTMLLoader-Objekt rendert die visuellen Elemente des HTML DOM als einzelnes Anzeigeobjekt. Sie können nicht direkt über die ActionScript-Anzeigelistenhierarchie auf die einzelnen Elemente des DOM zugreifen. Stattdessen greifen Sie mit der separaten DOM-API auf diese Elemente zu, die von HTMLLoader zur Verfügung gestellt werden.*

Grundlagen der Programmierung von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jede mit ActionScript 3.0 erstellte Anwendung hat eine Hierarchie von Anzeigeobjekten, die als *Anzeigeliste* bezeichnet wird (siehe unten). Die Anzeigeliste enthält alle sichtbaren Elemente der Anwendung.



Die Abbildung verdeutlicht, dass Anzeigeelemente in eine oder mehrere der folgenden Kategorien fallen:

- Bühne

  Die Bühne ist der grundlegende Container aller Anzeigeobjekte. Jede Anwendung verfügt über ein Stage-Objekt, das alle Anzeigeobjekte auf dem Bildschirm enthält. Die Bühne (Stage) ist der Container auf oberster Ebene und befindet sich an der Spitze der Anzeigelistenhierarchie:

  Jeder SWF-Datei ist eine ActionScript-Klasse zugeordnet, die als *Hauptklasse der SWF-Datei* bezeichnet wird. Wenn eine SWF-Datei in Flash Player oder Adobe AIR geöffnet wird, ruft Flash Player oder AIR die Konstruktorfunktion für diese Klasse auf. Die erstellte Instanz (die immer eine Art Anzeigeobjekt ist) wird dem Stage-Objekt als untergeordnetes Element hinzugefügt. Die Hauptklasse einer SWF-Datei erweitert immer die Sprite-Klasse (weitere Informationen finden Sie unter „Vorteile von Anzeigelisten“ auf Seite 167).

  Sie können über die `stage`-Eigenschaft jeder DisplayObject-Instanz auf die Bühne zugreifen. Weitere Informationen finden Sie unter „Festlegen der Stage-Eigenschaften“ auf Seite 176.

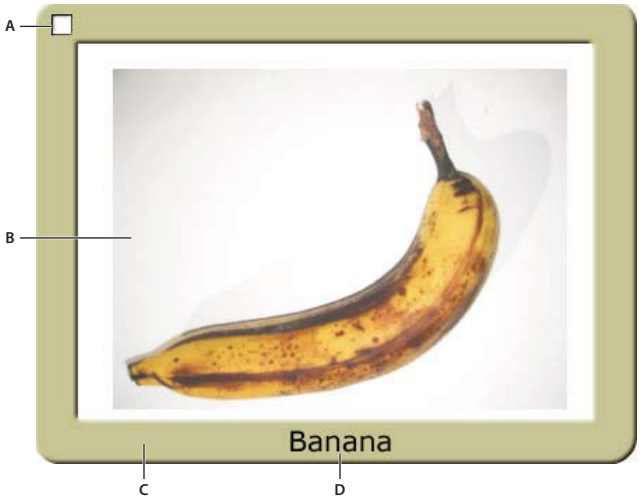
- Anzeigeobjekte

In ActionScript 3.0 sind alle Elemente, die in einer Anwendung auf dem Bildschirm erscheinen, Arten von *Anzeigeobjekten*. Das flash.display-Paket enthält eine DisplayObject-Klasse. Dies ist eine Basisklasse, die durch verschiedene andere Klassen erweitert wird. Diese verschiedenen Klassen stellen die unterschiedlichen Arten von Anzeigeobjekten wie Vektorformen, Movieclips und Textfelder dar, um nur einige zu nennen. Einen Überblick über diese Klassen finden Sie unter „Vorteile von Anzeigelisten“ auf Seite 167.

- Anzeigeobjektcontainer

  Anzeigeobjektcontainer sind besondere Anzeigeobjekte, die neben ihrer eigenen visuellen Darstellung auch untergeordnete Objekte enthalten können, bei denen es sich ebenfalls um Anzeigeobjekte handelt.

  Die DisplayObjectContainer-Klasse ist eine Unterklasse der DisplayObject-Klasse. Ein DisplayObjectContainer-Objekt kann mehrere Anzeigeobjekte in der *Child-Liste* (Liste der untergeordneten Elemente) enthalten. Die folgende Abbildung zeigt ein als „Sprite“ bezeichnetes DisplayObjectContainer-Objekt, das mehrere Anzeigeobjekte enthält:



A. *Ein SimpleButton-Objekt. Dieses Anzeigeobjekt hat verschiedene Zustände: „Auf“, „Gedrückt“ und „Darüber“.* ***B.*** *Ein Bitmap-Objekt. In diesem Fall wurde das Bitmap-Objekt über ein Loader-Objekt aus einer externen JPEG-Datei geladen.* ***C.*** *Ein Shape-Objekt. Der „Bilderrahmen“ enthält ein gerundetes Rechteck, das in ActionScript gezeichnet wurde. Auf dieses Shape-Objekt wurde ein Drop Shadow-Filter angewendet.* ***D.*** *Ein TextField-Objekt.*

  Im Kontext von Anzeigeobjekten werden DisplayObjectContainer-Objekte auch als *Anzeigeobjektcontainer* oder einfach als *Container* bezeichnet. Wie bereits erwähnt, ist die Bühne ein Anzeigeobjektcontainer.

  Obwohl alle sichtbaren Anzeigeobjekte von der DisplayObject-Klasse geerbt werden, ist der Objekttyp jedes Anzeigeobjekt eine spezielle Unterklasse der DisplayObject-Klasse. Beispielsweise gibt es für die Shape- oder die Video-Klasse eine Konstruktorfunktion, jedoch nicht für die DisplayObject-Klasse.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die Ihnen beim Programmieren von ActionScript-Grafiken häufig begegnen:

Alpha Die Farbwertdarstellung des Transparenzbetrags (genauer gesagt, des Opazitätsbetrags) in einer Farbe. Beispielsweise wird eine Farbe mit einem Alphakanal von 60 % nur mit 60 % der vollen Stärke und einer Transparenz von 40 % angezeigt.

Bitmapgrafik Eine Grafik, die auf einem Computer als Raster (aus Zeilen und Spalten) von farbigen Pixeln angezeigt wird. Im Allgemeinen werden digitale Fotos und ähnliche Bilder als Bitmapgrafiken angezeigt.

Mischmodus Eine Angabe, wie die Inhalte zweier überlappender Bilder miteinander interagieren sollen. Im Allgemeinen blockiert ein undurchsichtiges Bild ein darunter liegendes Bild so, dass es nicht sichtbar ist. Mit einem Mischmodus können jedoch die Farben der Bilder so miteinander verschmolzen werden, dass eine Kombination aus den beiden Bildern entsteht.

Anzeigeliste Die Hierarchie der Anzeigeobjekte, die als sichtbarer Inhalt von Flash Player und AIR auf dem Bildschirm angezeigt werden. Die Bühne ist der Stamm der Anzeigeliste. Zur Anzeigeliste gehören alle Anzeigeobjekte, die sich auf der Bühne befinden, oder ihre untergeordneten Elemente (auch dann, wenn ein Objekt nicht wirklich dargestellt wird, da es sich z. B. außerhalb der Bühnengrenzen befindet).

Anzeigeobjekt Ein Objekt, das einen bestimmten Typ eines sichtbaren Inhalts in Flash Player oder AIR darstellt. Es können nur Anzeigeobjekte in der Anzeigeliste enthalten sein und alle Anzeigeobjektklassen sind Unterklassen der DisplayObject-Klasse.

Anzeigeobjektcontainer Eine spezieller Typ eines Anzeigeobjekts, das neben der eigenen visuellen Darstellung noch untergeordnete Anzeigeobjekte enthalten kann.

Hauptklassen der SWF-Datei Die Klasse, die das Verhalten des am weitesten außen liegenden Anzeigeobjekts in einer SWF-Datei definiert, die begrifflich die Klasse für die SWF-Datei selbst ist. In einer in der Flash-Authoring-Umgebung erstellten SWF ist die Hauptklasse beispielsweise die Dokumentklasse. Sie hat eine „Hauptzeitleiste“, die alle anderen Zeitleisten enthält; die Hauptklasse der SWF-Datei ist die Klasse, von der die Hauptzeitleiste eine Instanz darstellt.

Maskieren Eine Technik zum Ausblenden von bestimmten Teilen eines Bildes (oder umgekehrt zum Einblenden nur von bestimmten Teilen eines Bildes). Die Teile des maskierten Bilds werden transparent, sodass der darunter liegende Inhalt hindurch scheint. Der englische Begriff (Masking) leitet sich vom englischen Wort für Abklebeband ab (masking tape), mit dem verhindert wird, das in bestimmten Bereichen Farbe aufgetragen wird.

Bühne Der sichtbare Container, der die Basis oder den Hintergrund für alle sichtbaren Inhalte einer SWF-Datei bildet.

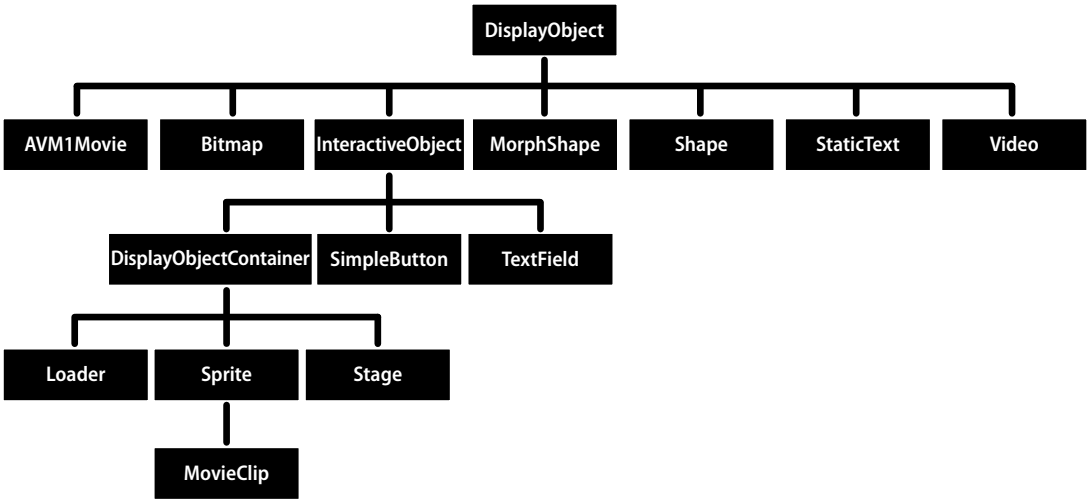
Transformation Ändern der visuellen Merkmale einer Grafik, beispielsweise durch Drehen, Ändern der Größe, Neigen, Verzerren oder Ändern der Farbe eines Objekts.

Vektorgrafik Eine Grafik, die auf einem Computer als Linien und Formen mit bestimmten Eigenschaften gezeichnet wird (z. B. Stärke, Länge, Größe, Winkel und Position).

Wichtige Anzeigeklassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das ActionScript 3.0-flash.display-Paket enthält Klassen für visuelle Objekte, die in Flash Player oder AIR angezeigt werden können. Die folgende Abbildung zeigt die Beziehungen der Unterklassen dieser Hauptanzeigeobjektklassen.



Die Abbildung zeigt die Klassenvererbung bei Anzeigeobjektklassen. Beachten Sie, dass sich einige dieser Klassen, insbesondere „StaticText“, „TextField“ und „Video“, nicht im flash.display-Paket befinden, aber dennoch von der DisplayObject-Klasse erben.

Alle Klassen, welche die DisplayObject-Klasse erweitern, erben ihre Methoden und Eigenschaften. Weitere Informationen finden Sie unter „Eigenschaften und Methoden der DisplayObject-Klasse“ auf Seite 170.

Sie können Objekte der folgenden, im flash.display-Paket enthaltenen Klassen instanziieren:

- Bitmap – Sie verwenden die Bitmap-Klasse zum Definieren von Bitmap-Objekten, die entweder aus externen Dateien geladen oder über ActionScript dargestellt werden. Bitmaps aus externen Dateien werden über die Loader-Klasse geladen. Sie können GIF-, JPG- oder PNG-Dateien laden. Sie können ein BitmapData-Objekt auch mit benutzerdefinierten Daten erstellen und dann ein Bitmap-Objekt erzeugen, das diese Daten verwendet. Zum Ändern von geladenen oder in ActionScript erstellten Bitmaps können Sie die Methoden der BitmapData-Klasse verwenden. Weitere Informationen finden Sie unter „Laden von Anzeigeobjekten“ auf Seite 212 und „Verwenden von Bitmaps“ auf Seite 258.
- Loader – Mit der Loader-Klasse können Sie externe Daten laden (entweder SWF-Dateien oder Grafiken). Weitere Informationen finden Sie unter „Dynamisches Laden von Anzeigeinhalten“ auf Seite 211.
- Shape – Mit der Shape-Klasse können Sie Vektorgrafiken wie beispielsweise Rechtecke, Linien, Kreise usw. erstellen. Weitere Informationen finden Sie unter „Verwenden der Zeichnungs-API“ auf Seite 236.
- SimpleButton – Ein SimpleButton-Objekt ist die ActionScript-Darstellung eines Schaltflächensymbols, das mit dem Flash-Authoring-Tool erstellt wurde. Eine SimpleButton-Instanz hat vier Schaltflächenzustände: „Auf“, „Gedrückt“, „Darüber“ und „Kollisionserkennung“ (der Bereich, der Maus- und Tastaturereignissen entspricht).
- Sprite – Ein Sprite-Objekt kann Grafiken von sich selbst und untergeordnete Anzeigeobjekte enthalten. (Die Sprite-Klasse erweitert die DisplayObjectContainer-Klasse). Weitere Informationen finden Sie unter „Arbeiten mit Anzeigeobjektcontainern“ auf Seite 170 und „Verwenden der Zeichnungs-API“ auf Seite 236.

- MovieClip – Ein MovieClip-Objekt ist die ActionScript-Form eines Movieclip-Symbols, das mit dem Flash-Authoring-Tool erstellt wurde. In der Praxis ähnelt ein Movieclip einem Sprite-Objekt, es hat nur zusätzlich eine Zeitleiste. Weitere Informationen finden Sie unter „Verwenden von Movieclips“ auf Seite 344.

Die folgenden Klassen, die nicht im flash.display-Paket enthalten sind, sind Unterklassen der DisplayObject-Klasse:

- Die TextField-Klasse, die sich im flash.text-Paket befindet, ist ein Anzeigeobjekt zur Darstellung von Text und Eingaben. Weitere Informationen finden Sie unter „Grundlagen der Textverarbeitung“ auf Seite 397.
- Die im flash.text.engine-Paket enthaltene TextLine-Klasse ist das Anzeigeobjekt, mit dem Textzeilen angezeigt werden, die von der Flash Text Engine und dem Text Layout Framework erstellt wurden. Weitere Informationen finden Sie unter „Verwenden der Flash Text Engine“ auf Seite 425 und „Verwenden des Text Layout Framework“ auf Seite 455.
- Die Video-Klasse, die sich im flash.media-Paket befindet, ist ein Anzeigeobjekt zur Darstellung von Videodateien. Weitere Informationen finden Sie unter „Verwenden von Videos“ auf Seite 505.

Die folgenden Klassen befinden sich im flash.display-Paket und erweitern die DisplayObject-Klasse, jedoch können Sie keine Instanzen dieser Klassen erstellen. Stattdessen dienen sie als übergeordnete Klassen für andere Anzeigeobjekte und führen allgemeine Funktionsmerkmale in einer Klasse zusammen.

- AVM1Movie – Die AVM1Movie-Klasse dient zur Darstellung von geladenen SWF-Dateien, die in ActionScript 1.0 und 2.0 erstellt wurden.
- DisplayObjectContainer – Die Klassen „Loader“, „Stage“, „Sprite“ und „MovieClip“ erweitern die DisplayObjectContainer-Klasse. Weitere Informationen finden Sie unter „Arbeiten mit Anzeigeobjektcontainern“ auf Seite 170.
- InteractiveObject – InteractiveObject ist die Basisklasse für alle Objekte, die für die Interaktion mit Maus und Tastatur verwendet werden. Die Objekte „SimpleButton“, „TextField“, „Loader“, „Sprite“, „Stage“ und „MovieClip“ sind Unterklassen der InteractiveObject-Klasse. Weitere Informationen zum Erstellen von Maus- und Tastaturinteraktion finden Sie unter „Grundlagen der Benutzerinteraktion“ auf Seite 592.
- MorphShape – Diese Objekte werden erstellt, wenn Sie einen Form-Tween in der Flash-Authoring-Umgebung erstellen. Sie können sie mit ActionScript nicht instanziieren, aber Sie können über die Anzeigeliste darauf zugreifen.
- Stage – Die Stage-Klasse erweitert die DisplayObjectContainer-Klasse. Es gibt eine Stage-Instanz für eine Anwendung und sie befindet sich an der Spitze der Anzeigelistenhierarchie: Sie können über die `stage`-Eigenschaft jeder DisplayObject-Instanz auf die Bühne zugreifen. Weitere Informationen finden Sie unter „Festlegen der Stage-Eigenschaften“ auf Seite 176.

Darüber hinaus wird die DisplayObject-Klasse durch die StaticText-Klasse aus dem flash.text-Paket erweitert, aber Sie können keine Instanz dieser Klasse im Code erstellen. Statische Textfelder werden nur in Flash erstellt.

Die folgenden Klassen sind weder Anzeigeobjekte noch Anzeigeobjektcontainer und werden nicht in der Anzeigeliste aufgeführt; sie zeigen jedoch Grafiken auf der Bühne an. Diese Klassen zeichnen Inhalt in ein als Viewport bezeichnetes Rechteck, das relativ zur Bühne positioniert wird.

- StageVideo – Die StageVideo-Klasse zeigt Videoinhalt an und verwendet dabei nach Möglichkeit die Hardwarebeschleunigung. Diese Klasse ist ab Flash Player 10.2 verfügbar. Weitere Informationen finden Sie unter „Verwenden der StageVideo-Klasse für die hardwarebeschleunigte Darstellung“ auf Seite 545.
- StageWebView – Die StageWebView-Klasse zeigt HTML-Inhalt an. Diese Klasse ist ab AIR 2.5 verfügbar. Weitere Informationen finden Sie unter „StageWebView-Objekte“ auf Seite 1098.

Die folgenden fl.display-Klassen bieten ähnliche Funktionalität wie die flash.display.Loader- und LoaderInfo-Klassen. Verwenden Sie diese Klassen anstelle der entsprechenden flash.display-Klassen, wenn Sie Inhalt in der Flash Professional-Umgebung (CS5.5 oder höher) entwickeln. In dieser Umgebung lassen sich mit diesen Klassen Probleme bezüglich TLF mit dem RSL-Vorausladen beheben. Weitere Informationen finden Sie unter „Verwenden der ProLoader- und ProLoaderInfo-Klassen“ auf Seite 216.

- fl.display.ProLoader – entspricht flash.display.Loader
- fl.display.ProLoaderInfo – entspricht flash.display.LoaderInfo

Vorteile von Anzeigelisten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 3.0 gibt es separate Klassen für die verschiedenen Anzeigeobjektarten. In ActionScript 1.0 und 2.0 waren viele dieser Objektarten in einer Klasse zusammengefasst: der MovieClip-Klasse.

Die Individualisierung von Klassen und die hierarchische Struktur von Anzeigelisten bieten zahlreiche Vorteile:

- Effizientere Darstellung und kleinere Dateien
- Verbesserte Tiefenverwaltung
- Vollständiges Durchlaufen der Anzeigeliste
- Anzeigeobjekte außerhalb der Liste
- Einfacheres Klassifizieren von Anzeigeobjekten in Untergruppen

Effizientere Darstellung und kleinere Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 1.0 und 2.0 konnten Sie Formen nur in einem MovieClip-Objekt zeichnen. In ActionScript 3.0 gibt es einfachere Anzeigeobjektklassen, in denen Sie Formen zeichnen können. Da diese ActionScript 3.0-Anzeigeobjektklassen keinen vollständigen Satz an Methoden und Eigenschaften wie ein MovieClip-Objekt enthalten, belegen sie weniger Speicher- und Prozessorressourcen.

Beispielsweise enthält jedes MovieClip-Objekt Eigenschaften für die Zeitleiste des Movieclips, ein Shape-Objekt jedoch nicht. Die Eigenschaften zur Verwaltung der Zeitleiste können einen Großteil der Speicher- und Prozessorressourcen belegen. In ActionScript 3.0 wird durch das Verwenden des Shape-Objekts eine bessere Performance erreicht. Das Shape-Objekt belastet den Speicher geringer als das komplexere MovieClip-Objekt. Flash Player und AIR müssen keine nicht verwendeten MovieClip-Eigenschaften verwalten. Dies erhöht die Geschwindigkeit und reduziert den Speicherbedarf des Objekts.

Verbesserte Tiefenverwaltung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 1.0 und 2.0 wurde die Tiefe über ein lineares Tiefenverwaltungsschema und Methoden wie z. B. `getNextHighestDepth()` verwaltet.

ActionScript 3.0 enthält die DisplayObjectContainer-Klasse, die einfacher anzuwendende Methoden und Eigenschaften zur Verwaltung der Anzeigeobjekttiefe bietet.

Wenn Sie in ActionScript 3.0 ein Anzeigeobjekt an eine neue Position in der Child-Liste einer DisplayObjectContainer-Instanz verschieben, werden die anderen Elemente der Child-Liste automatisch neu positioniert und erhalten entsprechende Child-Index-Positionen im Anzeigeobjektcontainer.

Darüber hinaus ist es in ActionScript 3.0 immer möglich, alle untergeordneten Objekte eines Anzeigeobjektcontainers zu erkennen. Jede DisplayObjectContainer-Instanz hat eine `numChildren`-Eigenschaft, in der die Anzahl der untergeordneten Elemente im Anzeigeobjektcontainer aufgeführt ist. Da die Child-Liste eines Anzeigeobjektcontainers immer eine Indexliste ist, können Sie jedes Objekt in der Liste von der ersten (0) bis zur letzten Indexposition (`numChildren - 1`) untersuchen. Dies war mit den Methoden und Eigenschaften eines MovieClip-Objekts in ActionScript 1.0 und 2.0 nicht möglich.

In ActionScript 3.0 können Sie die Anzeigeliste bequem sequenziell durchlaufen; es gibt keine Lücken bei den Indexpositionszahlen der Child-Liste eines Anzeigeobjektcontainers. Das Durchlaufen der Anzeigeliste und Verwalten der Objekttiefe ist viel einfacher als in ActionScript 1.0 und 2.0. In ActionScript 1.0 und 2.0 konnte ein Movieclip Objekte mit Lücken in der Tiefenebene enthalten, wodurch das Durchlaufen der Objektliste kompliziert wurde. In ActionScript 3.0 wird jede Child-Liste eines Anzeigeobjektcontainers intern als Array zwischengespeichert. Dadurch ist sehr schnelles Nachschlagen (über den Index) möglich. Auch das Durchlaufen alle untergeordneten Elemente eines Anzeigeobjektcontainers erfolgt sehr schnell.

In ActionScript 3.0 können Sie auch mit der `getChildByName()`-Methode der DisplayObjectContainer-Klasse auf die untergeordneten Elemente eines Anzeigeobjektcontainers zugreifen.

Vollständiges Durchlaufen der Anzeigeliste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 1.0 und 2.0 konnten Sie auf einige Objekte, wie z. B. in der Flash-Authoring-Umgebung gezeichnete Vektorformen, nicht zugreifen. In ActionScript 3.0 können Sie auf alle Objekte in der Anzeigeliste zugreifen, unabhängig davon, ob diese mit ActionScript oder in der Flash-Authoring-Umgebung erstellt wurden. Nähere Einzelheiten finden Sie unter „Durchlaufen der Anzeigeliste“ auf Seite 174.

Anzeigeobjekte außerhalb der Liste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 3.0 können Sie Anzeigeobjekte erstellen, die nicht in der sichtbaren Anzeigeliste erscheinen. Diese Objekte werden als *Offlist*-Anzeigeobjekte (außerhalb der Liste) bezeichnet. Ein Anzeigeobjekt wird nur dann der sichtbaren Anzeigeliste hinzugefügt, wenn Sie die `addChild()`- oder `addChildAt()`-Methode einer DisplayObjectContainer-Instanz aufrufen, die der Anzeigeliste bereits hinzugefügt wurde.

Mit diesen Offlist-Anzeigeobjekten können Sie komplexe Anzeigeobjekte zusammensetzen, beispielsweise Anzeigeobjekte mit mehreren Anzeigeobjektcontainern, die wiederum mehrere Anzeigeobjekte enthalten. Mit Offlist-Anzeigeobjekten können Sie komplizierte Objekte zusammensetzen, ohne Verarbeitungszeit zur Darstellung dieser Anzeigeobjekte aufzuwenden. Sie können ein Offlist-Anzeigeobjekt dann bei Bedarf einer Anzeigeliste hinzufügen. Außerdem können Sie ein untergeordnetes Element eines Anzeigeobjektcontainers zu einer Anzeigeliste hinzufügen oder daraus entfernen und an jede Position in der Anzeigeliste verschieben.

Einfacheres Klassifizieren von Anzeigeobjekten in Untergruppen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 1.0 und 2.0 mussten Sie neue MovieClip-Objekte in der Regel einer SWF-Datei hinzufügen, um grundlegende Formen zu erstellen oder Bitmaps anzuzeigen. In ActionScript 3.0 enthält die DisplayObject-Klasse viele integrierte Unterklassen, so z. B. „Shape“ und „Bitmap“. Da die Klassen in ActionScript 3.0 speziell für bestimmte Objekttypen entwickelt wurden, ist es einfacher, grundlegende Unterklassen der integrierten Klassen zu erstellen.

Um in ActionScript 2.0 einen Kreis zu zeichnen, können Sie eine CustomCircle-Klasse erstellen, welche die MovieClip-Klasse erweitert, wenn ein Objekt der benutzerdefinierten Klasse instanziiert wird. Jedoch würde diese Klasse zahlreiche Eigenschaften und Methoden von der MovieClip-Klasse übernehmen (beispielsweise `totalFrames`), die für diese Klasse nicht gelten. In ActionScript 3.0 erstellen Sie einfach eine CustomCircle-Klasse, die das Shape-Objekt erweitert und somit keine unnötigen Eigenschaften und Methoden übernimmt, die in der MovieClip-Klasse enthalten sind. Der folgende Code zeigt ein Beispiel einer CustomCircle-Klasse:

```
import flash.display.*;

public class CustomCircle extends Shape
{
    var xPos:Number;
    var yPos:Number;
    var radius:Number;
    var color:uint;
    public function CustomCircle(xInput:Number,
                                 yInput:Number,
                                 rInput:Number,
                                 colorInput:uint)
    {
        xPos = xInput;
        yPos = yInput;
        radius = rInput;
        color = colorInput;
        this.graphics.beginFill(color);
        this.graphics.drawCircle(xPos, yPos, radius);
    }
}
```

Verwenden von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Nachdem Sie sich mit den allgemeinen Konzepten „Bühne“, „Anzeigeobjekt“, „Anzeigeobjektcontainer“ und „Anzeigeliste“ vertraut gemacht haben, finden Sie nun in diesem Abschnitt weitere Informationen zum Verwenden von Anzeigeobjekten in ActionScript 3.0.

Eigenschaften und Methoden der DisplayObject-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Alle Anzeigeobjekte sind Unterklassen der DisplayObject-Klasse und als solche erben sie die Eigenschaften und Methoden von ihr. Die geerbten Eigenschaften sind die grundlegenden Eigenschaften, die für alle Anzeigeobjekte gelten. Beispielsweise hat jedes Anzeigeobjekt eine Eigenschaft `x` und eine Eigenschaft `y`, mit denen die Objektposition im Anzeigeobjektcontainer festgelegt wird.

Mit dem DisplayObject-Klassenkonstruktor können Sie keine DisplayObject-Instanz erstellen. Um ein Objekt mit dem `new`-Operator zu instanziieren, müssen Sie einen weiteren Objekttyp erstellen (ein Objekt, das eine Unterklasse der DisplayObject-Klasse ist), beispielsweise ein Sprite. Auch wenn Sie eine benutzerdefinierte Anzeigeobjektklasse erstellen möchten, müssen Sie eine Unterklasse einer der Anzeigeobjekt-Unterklassen erstellen, die über eine verwendbare Konstruktorfunktion verfügt (beispielsweise die Shape- oder die Sprite-Klasse). Weitere Informationen finden Sie in der Beschreibung der DisplayObject-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Hinzufügen von Anzeigeobjekten zur Anzeigeliste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie ein Anzeigeobjekt instanziieren, erscheint es erst dann auf dem Bildschirm (auf der Bühne), wenn Sie die Anzeigeobjektinstanz einem Anzeigeobjektcontainer hinzufügen, der sich bereits in der Anzeigeliste befindet. Im folgenden Beispielcode würde das TextField-Objekt `myText` nicht sichtbar sein, wenn Sie die letzte Codezeile weglassen. In der letzten Codezeile muss das Schlüsselwort `this` auf den Anzeigeobjektcontainer verweisen, der bereits zur Anzeigeliste hinzugefügt ist.

```
import flash.display.*;
import flash.text.TextField;
var myText:TextField = new TextField();
myText.text = "Buenos dias.";
this.addChild(myText);
```

Wenn Sie der Bühne ein visuelles Element hinzufügen, wird dieses Element zu einem *untergeordneten Element* (Englisch „child“) des Stage-Objekts. Die erste SWF-Datei, die in einer Anwendung geladen wird (beispielsweise eine Datei, die Sie in eine HTML-Seite einbetten) wird automatisch als untergeordnetes Element der Bühne hinzugefügt. Es kann sich um ein Objekt beliebigen Typs handeln, das die Sprite-Klasse erweitert.

Alle Anzeigeobjekte, die Sie *ohne* ActionScript erstellen – beispielsweise durch Hinzufügen eines MXML-Tags in einer Flex-MXML-Datei oder durch Platzieren eines Objekts auf der Bühne in Flash Professional – werden der Anzeigeliste hinzugefügt. Obwohl Sie diese Anzeigeobjekte nicht mit ActionScript hinzugefügt haben, können Sie über ActionScript darauf zugreifen. So stellt der folgende Code die Breite eines Objekts namens `button1` ein, das in der Authoring-Umgebung hinzugefügt wurde (nicht mit ActionScript):

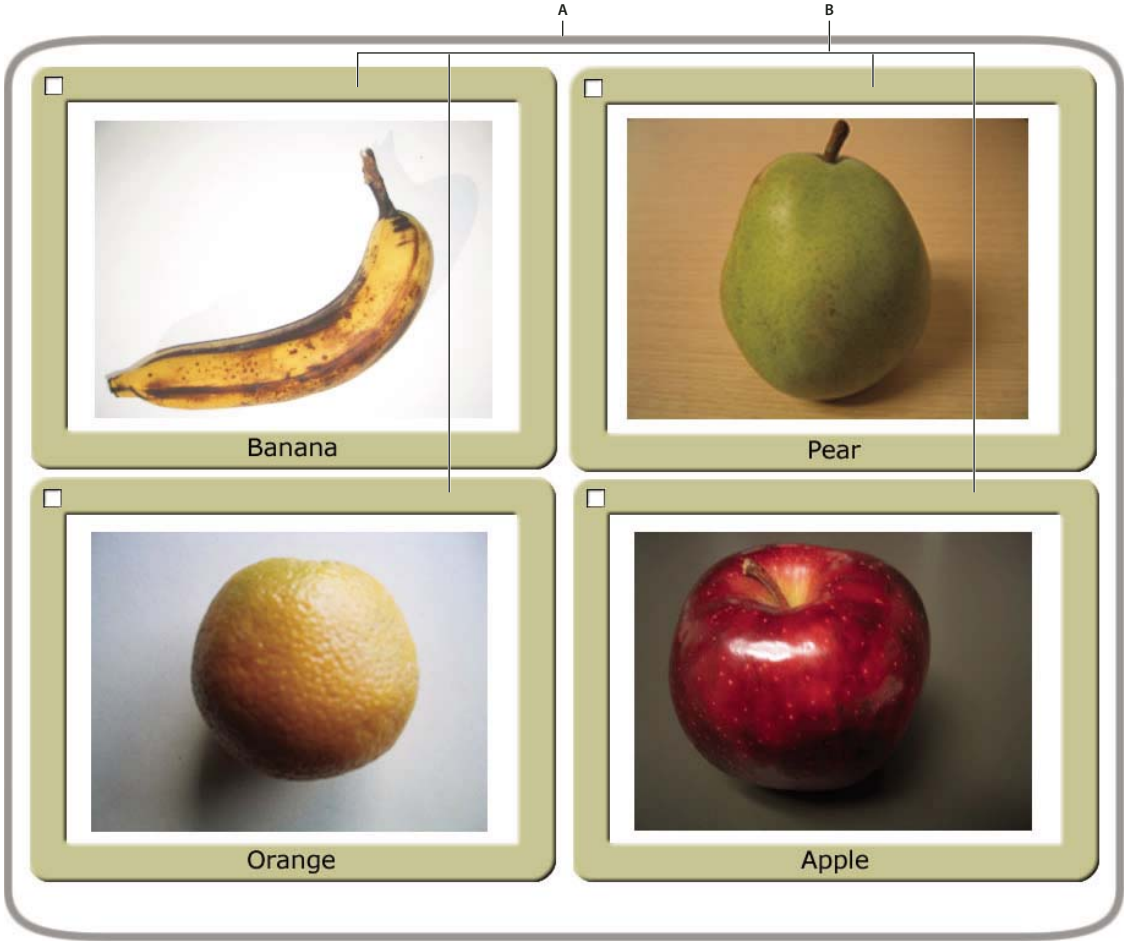
```
button1.width = 200;
```

Arbeiten mit Anzeigeobjektcontainern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn ein DisplayObjectContainer-Objekt aus der Anzeigeliste gelöscht, verschoben oder auf andere Weise transformiert wurde, werden die entsprechenden Anzeigeobjekte im Anzeigeobjektcontainer ebenfalls gelöscht, verschoben oder transformiert.

Ein Anzeigeobjektcontainer ist selbst eine Art Anzeigeobjekt – es kann einem anderen Anzeigeobjektcontainer hinzugefügt werden. Das folgende Bild zeigt beispielsweise einen Anzeigeobjektcontainer, `pictureScreen`, der eine Konturform und vier weitere Anzeigeobjektcontainer (des Typs „PictureFrame“) enthält:



A. *Eine Form definiert den Rahmen des pictureScreen-Anzeigeobjektcontainers.* ***B.*** *Vier Anzeigeobjektcontainer, bei denen es sich um untergeordnete Elemente des pictureScreen-Objekts handelt.*

Damit ein Anzeigeobjekt in der Anzeigeliste erscheint, muss es einem Anzeigeobjektcontainer hinzugefügt werden, der sich bereits in der Anzeigeliste befindet. Dazu verwenden Sie die `addChild()`- oder die `addChildAt()`-Methode des Containerobjekts. Ohne die letzte Zeile des folgenden Codes wird das Objekt `myTextField` nicht angezeigt:

```
var myTextField:TextField = new TextField();
myTextField.text = "hello";
this.root.addChild(myTextField);
```

In diesem Codebeispiel verweist `this.root` auf den MovieClip-Anzeigeobjektcontainer, der den Code enthält. In Ihrem Code können Sie einen anderen Container angeben.

Verwenden Sie die `addChildAt()`-Methode, um das untergeordnete Element an einer bestimmten Position in der Child-Liste des Anzeigeobjektcontainers hinzuzufügen. Diese nullbasierten Indexpositionen in der Child-Liste beziehen sich auf die Ebenen der Anzeigeobjekte (von vorne nach hinten). Betrachten Sie das Beispiel der folgenden drei Anzeigeobjekte. Jedes Objekt wurde aus einer benutzerdefinierten Klasse namens „Ball“ erstellt.

Die Reihenfolge der Ebenen dieser Anzeigeobjekte in ihrem Container kann mithilfe der Methode `addChildAt()` eingestellt werden. Betrachten Sie den folgenden Beispielcode:

```
ball_A = new Ball(0xFFCC00, "a");
ball_A.name = "ball_A";
ball_A.x = 20;
ball_A.y = 20;
container.addChild(ball_A);

ball_B = new Ball(0xFFCC00, "b");
ball_B.name = "ball_B";
ball_B.x = 70;
ball_B.y = 20;
container.addChild(ball_B);

ball_C = new Ball(0xFFCC00, "c");
ball_C.name = "ball_C";
ball_C.x = 40;
ball_C.y = 60;
container.addChildAt(ball_C, 1);
```

Nach dem Ausführen dieses Codes sind die Anzeigeobjekte im DisplayObjectContainer-Objekt `container` in der folgenden Reihenfolge positioniert. Beachten Sie die Ebenenanordnung der Objekte.

Um ein Objekt auf der obersten Ebene in der Anzeigeliste zu positionieren, fügen Sie es der Liste einfach noch einmal hinzu. Verwenden Sie beispielsweise die folgende Zeile, um `ball_A` nach dem vorangegangenen Code in die oberste Ebene des Stapels zu verschieben:

```
container.addChild(ball_A);
```

Dieser Code löscht `ball_A` an seiner Position in der Anzeigeliste von `container` und fügt es an der Spitze der Liste neu hinzu – dies führt letztlich dazu, dass das Objekt in die oberste Ebene des Stapels verschoben wird.

Mit der `getChildAt()`-Methode können Sie die Ebenenreihenfolge der Anzeigeobjekte überprüfen. Die `getChildAt()`-Methode gibt die untergeordneten Objekte in einem Container basierend auf der Indexnummer zurück, die Sie übergeben haben. Im folgenden Beispielcode werden die Namen der Anzeigeobjekte an den verschiedenen Positionen in der Child-Liste des DisplayObjectContainer-Objekts `container` sichtbar gemacht:

```
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_C
trace(container.getChildAt(2).name); // ball_B
```

Wenn Sie ein Anzeigeobjekt aus der Child-Liste des übergeordneten Containers entfernen, werden die höheren Elemente in der Liste um jeweils eine Position in der untergeordneten Indexposition nach unten verschoben. Im folgenden Beispiel setzen wir den Code aus den vorangegangenen Beispiel fort. Es zeigt, wie ein Anzeigeobjekt, das sich an Position 2 im DisplayObjectContainer-Objekt `container` befand, an die Position 1 verschoben wird, wenn ein niedrigeres Anzeigeobjekt aus der Child-Liste entfernt wird:

```
container.removeChild(ball_C);
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_B
```

Die Methoden `removeChild()` und `removeChildAt()` löschen eine Anzeigeobjektinstanz nicht vollständig. Sie entfernen sie einfach nur aus der Child-Liste des Containers. Die Instanz kann dennoch von einer anderen Variablen für Verweise genutzt werden. (Verwenden Sie den Operator `delete`, um ein Objekt vollständig zu löschen.)

Da ein Anzeigeobjekt nur einen übergeordneten Container hat, können Sie eine Instanz eines Anzeigeobjekts nur einem Anzeigeobjektcontainer hinzufügen. Im folgenden Beispielcode wird gezeigt, dass das Anzeigeobjekt `tf1` nur in einem Container existieren kann (in diesem Fall ein Sprite, das die DisplayObjectContainer-Klasse erweitert):

```
tf1:TextField = new TextField();
tf2:TextField = new TextField();
tf1.name = "text 1";
tf2.name = "text 2";

container1:Sprite = new Sprite();
container2:Sprite = new Sprite();

container1.addChild(tf1);
container1.addChild(tf2);
container2.addChild(tf1);

trace(container1.numChildren); // 1
trace(container1.getChildAt(0).name); // text 2
trace(container2.numChildren); // 1
trace(container2.getChildAt(0).name); // text 1
```

Wenn Sie ein Anzeigeobjekt, das in einem Anzeigeobjektcontainer enthalten ist, einem anderen Anzeigeobjektcontainer hinzufügen, wird es aus der Child-Liste des ersten Anzeigeobjektcontainers entfernt.

Neben den beiden oben beschriebenen Methoden definiert die DisplayObjectContainer-Klasse weitere Methoden zum Arbeiten mit untergeordneten Anzeigeobjekten. Hierzu gehören u. a. die Folgenden:

- `contains()`: Ermittelt, ob ein Anzeigeobjekt ein untergeordnetes Element eines DisplayObjectContainer ist.
- `getChildByName()`: Ruft ein Anzeigeobjekt über den Namen ab.
- `getChildIndex()`: Gibt die Indexposition eines Anzeigeobjekts zurück.
- `setChildIndex()`: Ändert die Position eines untergeordneten Anzeigeobjekts.
- `removeChildren()`: Entfernt mehrere untergeordnete Anzeigeobjekte.

- `swapChildren()`: Kehrt die Reihenfolge zweier Anzeigeobjekte um (von vorne nach hinten).
- `swapChildrenAt()`: Kehrt die Reihenfolge zweier Anzeigeobjekte um (von vorne nach hinten), angegeben durch deren Indexwerte.

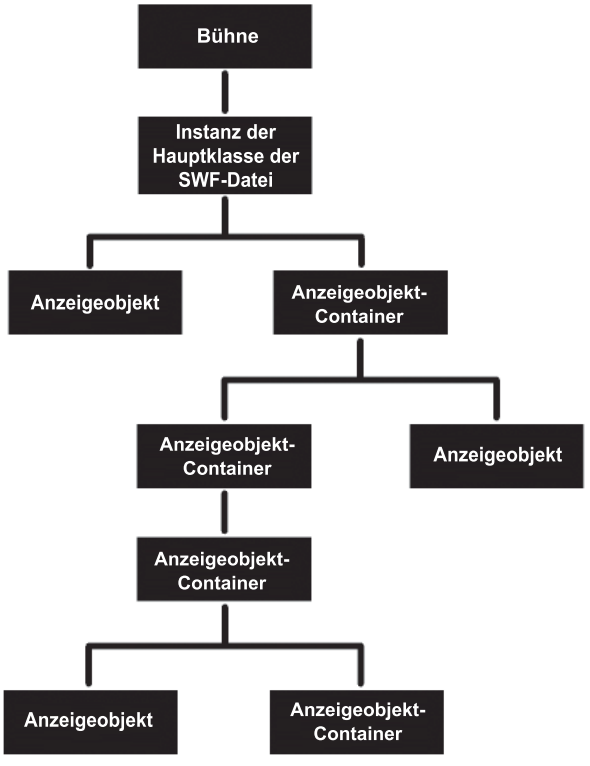
Weitere Informationen finden Sie in den jeweiligen Einträgen im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Zur Erinnerung: Ein Anzeigeobjekt, das sich nicht in der Anzeigeliste befindet (eines, das sich nicht in einem Anzeigeobjektcontainer befindet, der ein untergeordnetes Objekt der Bühne ist), wird als *Offlist*-Anzeigeobjekt bezeichnet.

Durchlaufen der Anzeigeliste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie Sie bereits gesehen haben, lässt sich die Anzeigeliste als eine Baumstruktur darstellen. Die Bühne, die mehrere Anzeigeobjekte enthalten kann, bildet den Stamm. Diese Anzeigeobjekte, die selbst Anzeigeobjektcontainer sind, können andere Anzeigeobjekte oder Anzeigeobjektcontainer enthalten.



Die DisplayObjectContainer-Klasse enthält Eigenschaften und Methoden zum Durchlaufen der Anzeigeliste mithilfe der Child-Listen der Anzeigeobjektcontainer. Im folgenden Beispielcode werden zwei Anzeigeobjekte, `title` und `pict`, zum `container`-Objekt hinzugefügt (ein Sprite, und die Sprite-Klasse erweitert die DisplayObjectContainer-Klasse):

```
var container:Sprite = new Sprite();
var title:TextField = new TextField();
title.text = "Hello";
var pict:Loader = new Loader();
var url:URLRequest = new URLRequest("banana.jpg");
pict.load(url);
pict.name = "banana loader";
container.addChild(title);
container.addChild(pict);
```

Die `getChildAt()`-Methode gibt die untergeordneten Elemente der Anzeigeliste an einer bestimmten Indexposition zurück:

```
trace(container.getChildAt(0) is TextField); // true
```

Sie können auch über den Namen auf die untergeordneten Objekte zugreifen. Jedes Anzeigeobjekt verfügt über eine Namenseigenschaft. Wenn Sie diese Eigenschaft nicht zuweisen, weist Flash Player einen Standardwert wie z. B. „`instance1"` zu. Im folgenden Beispielcode wird gezeigt, wie mit der `getChildByName()`-Methode auf ein untergeordnetes Anzeigeobjekt namens „`banana loader"` zugegriffen wird:

```
trace(container.getChildByName("banana loader") is Loader); // true
```

Die `getChildByName()`-Methode führt jedoch eventuell zu einer schlechteren Leistung als die `getChildAt()`-Methode.

Da ein Anzeigeobjektcontainer andere Anzeigeobjektcontainer als untergeordnete Objekte in seiner Anzeigeliste enthalten kann, können Sie die vollständige Anzeigeliste der Anwendung als eine Baumstruktur durchlaufen. Sobald in dem bereits vorgestellten Codeausschnitt der Ladevorgang für das `pict`-Loader-Objekt abgeschlossen ist, hat das `pict`-Objekt ein untergeordnetes Anzeigeobjekt geladen (die Bitmap). Für den Zugriff auf dieses Bitmap-Anzeigeobjekt können Sie `pict.getChildAt(0)` verwenden. Sie können auch `container.getChildAt(0).getChildAt(0)` schreiben (da `container.getChildAt(0) == pict`).

Die folgende Funktion liefert eine eingerückte `trace()`-Ausgabe der Anzeigeliste eines Anzeigeobjektcontainers:

```
function traceDisplayList(container:DisplayObjectContainer,indentString:String = ""):void
{
    var child:DisplayObject;
    for (var i:uint=0; i < container.numChildren; i++)
    {
        child = container.getChildAt(i);
        trace(indentString, child, child.name);
        if (container.getChildAt(i) is DisplayObjectContainer)
        {
            traceDisplayList(DisplayObjectContainer(child), indentString + "")
        }
    }
}
```

Adobe Flex

Wenn Sie Flex verwenden, sollten Sie wissen, dass Flex viele Komponenten-Anzeigeobjektklassen definiert und dass diese Klassen die Zugriffsmethoden für Anzeigelisten der DisplayObjectContainer-Klasse überschreiben. So überschreibt z. B. die Container-Klasse des mx.core-Pakets die `addChild()`-Methode und andere Methoden der DisplayObjectContainer-Klasse (die die Container-Klasse erweitert). Im Fall der `addChild()`-Methode überschreibt die Klasse die Methode dergestalt, dass nicht alle Typen von Anzeigeobjekten zu einer Container-Instanz in Flex hinzugefügt werden können. Die überschriebene Methode fordert in diesem Fall, dass das untergeordnete Objekt, das Sie hinzufügen, ein Typ des mx.core.UIComponent-Objekts ist.

Festlegen der Stage-Eigenschaften

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Stage-Klasse überschreibt die meisten Eigenschaften und Methoden der DisplayObject-Klasse. Wenn Sie eine diese überschriebenen Eigenschaften oder Methoden aufrufen, lösen Flash Player und AIR eine Ausnahme aus. Beispielsweise hat das Stage-Objekt keine `x`- oder `y`-Eigenschaften, da seine Position im Hauptcontainer der Anwendung festgelegt ist. Die `x`- und `y`-Eigenschaften beziehen sich auf die Position eines Anzeigeobjekts relativ zu seinem Container, und da die Bühne in keinem anderen Anzeigeobjektcontainer enthalten ist, treffen diese Eigenschaften nicht zu.

Hinweis: *Einige Eigenschaften und Methoden der Stage-Klasse stehen nur Anzeigeobjekten zur Verfügung, die sich in der gleichen Sicherheits-Sandbox wie die erste geladene SWF-Datei befinden. Nähere Einzelheiten finden Sie unter „Sicherheit der Bühne“ auf Seite 1139.*

Einstellen der Wiedergabebildrate

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `frameRate`-Eigenschaft der Stage-Klasse dient zum Einstellen der Bildrate für alle SWF-Dateien, die in die Anwendung geladen werden. Weitere Informationen finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Steuern der Bühnenskalierung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn eine Größenänderung an dem Teil des Bildschirms, der Flash Player oder AIR darstellt, durchgeführt wird, wird der Bühneninhalt von der Laufzeitumgebung automatisch angepasst. Die `scaleMode`-Eigenschaft der Stage-Klasse legt fest, wie der Bühneninhalt angepasst wird. Für diese Eigenschaft können vier verschiedene Werte festgelegt werden, die als Konstanten in der flash.display.StageScaleMode-Klasse definiert sind:

- `StageScaleMode.EXACT_FIT` skaliert die SWF so, dass sie die Bühne mit den neuen Abmessungen ausfüllt. Das ursprüngliche Seitenverhältnis des Inhalts wird dabei nicht berücksichtigt. Die Skalierungsfaktoren sind für Breite und Höhe möglicherweise nicht identisch; deshalb kann der Inhalt gestaucht oder gedehnt aussehen, wenn das Seitenverhältnis der Bühne geändert wird.
- `StageScaleMode.SHOW_ALL` skaliert die SWF so, dass sie vollständig in die neuen Abmessungen der Bühne passt. Das Seitenverhältnis des Inhalts wird dabei nicht geändert. In diesem Skalierungsmodus wird zwar der ganze Inhalt angezeigt, es können jedoch Letterbox-Ränder entstehen, wie die schwarzen Ränder beim Betrachten eines Films im Breitbildformat auf einem Standardfernseher.

- `StageScaleMode.NO_BORDER` skaliert die SWF so, dass sie die Bühne mit den neuen Abmessungen vollständig ausfüllt. Das Seitenverhältnis des Inhalts wird dabei nicht geändert. Bei diesem Skalierungsmodus wird der Anzeigebereich der Bühne voll ausgenutzt, das Bild kann jedoch abgeschnitten werden.
- `StageScaleMode.NO_SCALE` skaliert die SWF nicht. Wenn die neuen Abmessungen der Bühne kleiner sind, wird der Inhalt abgeschnitten. Sind die neuen Abmessungen größer, ist der zusätzliche Platz leer.

  Nur im `StageScaleMode.NO_SCALE`-Skalierungsmodus können die Eigenschaften `stageWidth` und `stageHeight` der Stage-Klasse verwendet werden, um die tatsächlichen Abmessungen (in Pixel) der Bühne nach der Größenänderung zu ermitteln. (In den anderen Skalierungsmodi geben die Eigenschaften `stageWidth` und `stageHeight` immer die ursprüngliche Breite und Höhe der SWF-Datei an.) Wenn `scaleMode` den Wert `StageScaleMode.NO_SCALE` aufweist, wird zusätzlich bei Größenänderungen der SWF-Datei das `resize`-Ereignis der Stage-Klasse ausgelöst, um entsprechende Anpassungen zu ermöglichen.

  Infolgedessen haben Sie im Modus `StageScaleMode.NO_SCALE` für `scaleMode` bei Bedarf eine größere Kontrolle darüber, wie Bildschirminhalte an Fenster angepasst werden, deren Größe sich geändert hat. Beispielsweise empfiehlt es sich für eine SWF-Datei mit einem Video und einer Steuerungsleiste, dass die Größe der Steuerungsleiste nach einer Größenänderung der Bühne gleich bleibt und nur die Größe des Videofensters an die geänderte Größe der Bühne angepasst wird. Dies wird im folgenden Beispiel veranschaulicht:

```
// mainContent is a display object containing the main content;
// it is positioned at the top-left corner of the Stage, and
// it should resize when the SWF resizes.

// controlBar is a display object (e.g. a Sprite) containing several
// buttons; it should stay positioned at the bottom-left corner of the
// Stage (below mainContent) and it should not resize when the SWF
// resizes.

import flash.display.Stage;
import flash.display.StageAlign;
import flash.display.StageScaleMode;
import flash.events.Event;

var swfStage:Stage = mainContent.stage;
swfStage.scaleMode = StageScaleMode.NO_SCALE;
swfStage.align = StageAlign.TOP_LEFT;
swfStage.addEventListener(Event.RESIZE, resizeDisplay);

function resizeDisplay(event:Event):void
{
    var swfWidth:int = swfStage.stageWidth;
    var swfHeight:int = swfStage.stageHeight;

    // Resize the main content area
    var newContentHeight:Number = swfHeight - controlBar.height;
    mainContent.height = newContentHeight;
    mainContent.scaleX = mainContent.scaleY;

    // Reposition the control bar.
    controlBar.y = newContentHeight;
}
```

Festlegen des Skalierungsmodus für die Bühne in AIR-Fenstern

Die `scaleMode`-Eigenschaft der Bühne bestimmt, wie die Bühne skaliert wird, und schneidet untergeordnete Anzeigeobjekte zu, wenn die Größe eines Fensters geändert wird. Nur der `noScale`-Modus sollte in AIR verwendet werden. In diesem Modus wird die Bühne nicht skaliert. Stattdessen ändert sich die Bühnengröße direkt mit den Fenstergrenzen. Wenn das Fenster verkleinert wird, werden Objekte möglicherweise entsprechend zugeschnitten.

Die Skalierungsmodi für die Bühne sind für Umgebungen vorgesehen, in denen Sie nicht immer Kontrolle über die Größe oder das Seitenverhältnis der Bühne haben, beispielsweise in einem Webbrowser. Mit diesen Modi können Sie den optimalen Kompromiss auswählen, wenn die Bühne nicht der idealen Größe oder dem idealen Seitenverhältnis der Anwendung entspricht. In AIR können Sie die Bühne immer steuern. Deshalb erzielen Sie in den meisten Fällen bessere Ergebnisse, wenn Sie das Layout des Inhalts ändern oder die Fensterabmessungen anpassen, als wenn Sie die Bühnenskalierung aktivieren.

Im Browser und für das anfängliche AIR-Fenster wird die Beziehung zwischen der Fenstergröße und dem anfänglichen Skalierungsfaktor aus der geladenen SWF-Datei gelesen. Wenn Sie jedoch ein NativeWindow-Objekt erstellen, wählt AIR eine zufällige Beziehung zwischen der Fenstergröße und dem Skalierungsfaktor von 72:1. Wenn Ihr Fenster eine Größe von 72 x 72 Pixel hat, wird deshalb ein 10 x 10 großes Rechteck mit der richtigen Größe von 10 x 10 Pixeln um das Fenster gezeichnet. Hat das Fenster dagegen eine Größe von 144 x 144 Pixeln, wird ein 10 x 10 Pixel großes Rechteck auf 20 x 20 Pixel skaliert. Wenn Sie als `scaleMode` unbedingt einen anderen Modus als `noScale` für eine Fensterbühne verwenden möchten, können Sie dies ausgleichen, indem Sie den Skalierungsfaktor für Anzeigeobjekte im Fenster auf ein Verhältnis von 72 Pixel relativ zur aktuellen Breite und Höhe der Bühne einstellen. Mit dem folgenden Code wird beispielsweise der erforderliche Skalierungsfaktor für ein Anzeigeobjekt namens `client` berechnet:

```
if(newWindow.stage.scaleMode != StageScaleMode.NO_SCALE){
client.scaleX = 72/newWindow.stage.stageWidth;
client.scaleY = 72/newWindow.stage.stageHeight;
}
```

Hinweis: *Für Flex- und HTML-Fenster wird `scaleMode` für die Bühne automatisch auf `noScale` eingestellt. Eine Änderung von `scaleMode` behindert die automatischen Layoutmechanismen, die in diesen Fenstertypen verwendet werden.*

Verwenden des Vollbildmodus

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Im Vollbildmodus können Sie die Bühne eines Movies so einrichten, dass sie den gesamten Monitor eines Benutzers füllt und keine Containerrahmen oder Menüs angezeigt werden. Mit der Eigenschaft `displayState` der Stage-Klasse wird der Vollbildmodus für eine SWF-Datei aktiviert bzw. deaktiviert. Die Eigenschaft `displayState` kann auf einen der Werte eingestellt werden, die durch die Konstanten in der flash.display.StageDisplayState-Klasse definiert sind. Um in den Vollbildmodus zu wechseln, setzen Sie die `displayState`-Eigenschaft auf `StageDisplayState.FULL_SCREEN`:

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

Um den interaktiven Vollbildmodus (neu in Flash Player 11.3) einzuschalten, stellen Sie die `displayState`-Eigenschaft auf `StageDisplayState.FULL_SCREEN_INTERACTIVE` ein:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

In Flash Player kann der Vollbildmodus durch ActionScript nur als Reaktion auf einen Mausklick (einschließlich Rechtsklick) oder einen Tastenanschlag initiiert werden. Bei AIR-Inhalt, der in der Sicherheits-Sandbox der Anwendung ausgeführt wird, muss der Vollbildmodus nicht als Reaktion auf eine Benutzeraktion aufgerufen werden.

Um den Vollbildmodus zu beenden, setzen Sie `displayState` auf `StageDisplayState.NORMAL`.

```
stage.displayState = StageDisplayState.NORMAL;
```

Darüber hinaus kann ein Benutzer den Vollbildmodus beenden, indem er den Fokus auf ein anderes Fenster setzt oder einen der folgenden Tastaturbefehle verwendet: Esc-Taste (alle Plattformen), Strg-W (Windows), Befehl-W (Mac) oder Alt-F4 (Windows).

Aktivieren des Vollbildmodus in Flash Player

Um den Vollbildmodus für eine SWF-Datei zu aktivieren, die in eine HTML-Seite eingebettet ist, muss der HTML-Code zum Einbetten von Flash Player ein `param`-Tag und ein `embed`-Attribut mit dem Namen `allowFullScreen` und dem Wert `true` enthalten. Beispiel:

```
<object>
    ...
    <param name="allowFullScreen" value="true" />
    <embed ... allowFullScreen="true" />
</object>
```

Wählen Sie in der Flash-Authoring-Umgebung „Datei“ > „Einstellungen für Veröffentlichungen“ und im Dialogfeld „Einstellungen für Veröffentlichungen“ auf der Registerkarte „HTML“ die Vorlage „Nur Flash - Vollbild zulassen“.

Stellen Sie in Flex sicher, dass die HTML-Vorlage `<object>`- und `<embed>`-Tags enthält, die den Vollbildmodus unterstützen.

Wenn Sie JavaScript in einer Webseite verwenden, um die SWF-eingebetteten Tags zu erzeugen, müssen Sie den JavaScript-Code ändern, um das Tag und das Attribut für den `allowFullScreen`-Parameter hinzuzufügen. Wenn Ihre HTML-Seite z. B. die `AC_FL_RunContent()`-Funktion verwendet (die in von Flash Professional und Flash Builder generierten HTML-Seiten verwendet wird), müssen Sie den `allowFullScreen`-Parameter wie folgt zum Funktionsaufruf hinzufügen:

```
AC_FL_RunContent(
    ...
    'allowFullScreen','true',
    ...
    ); //end AC code
```

Dies gilt nicht für SWF-Dateien, die im eigenständigen Flash Player ausgeführt werden.

Hinweis: *Wenn Sie den Fenstermodus („wmode“ in HTML) auf „Undurchsichtig ohne Fenster“ (opaque) oder „Durchsichtig ohne Fenster“ (transparent) einstellen, ist der Vollbildmodus immer undurchsichtig*

Es gibt auch sicherheitsbezogene Einschränkungen für die Verwendung des Vollbildmodus mit Flash Player in einem Browser. Diese Einschränkungen werden unter „Sicherheit“ auf Seite 1115 beschrieben.

Aktivieren des interaktiven Vollbildmodus in Flash Player 11.3 und höher

Flash Player 11.3 und höher unterstützt den interaktiven Vollbildmodus, in dem alle Tasten der Tastatur unterstützt werden mit Ausnahme von **Esc**, womit der interaktive Vollbildmodus beendet wird. Der interaktive Vollbildmodus ist hilfreich beim Gaming (zum Beispiel, um in einem Multiplayer-Spiel zu chatten, oder für WASD-Tastatursteuerung in einem Egoshooter).

Um den interaktiven Vollbildmodus für eine SWF-Datei zu aktivieren, die in eine HTML-Seite eingebettet ist, muss der HTML-Code zum Einbetten von Flash Player ein `param`-Tag und ein `embed`-Attribut mit dem Namen `allowFullScreenInteractive` und dem Wert `true` enthalten. Beispiel:

```
<object>
    ...
    <param name="allowFullScreenInteractive" value="true" />
    <embed ... allowFullScreenInteractive="true" />
</object>
```

Wählen Sie in der Flash-Authoring-Umgebung „Datei“ > „Einstellungen für Veröffentlichungen“ und im Dialogfeld „Einstellungen für Veröffentlichungen“ auf der Registerkarte „HTML“ die Vorlage „Nur Flash - Interaktives Vollbild zulassen“.

Stellen Sie in Flash Builder und Flex sicher, dass die HTML-Vorlagen die Tags `<object>` und `<embed>` enthalten, die den interaktiven Vollbildmodus unterstützen.

Wenn Sie JavaScript in einer Webseite verwenden, um die SWF-eingebetteten Tags zu erzeugen, müssen Sie den JavaScript-Code ändern, um das Tag und das Attribut für den `allowFullScreenInteractive` -Parameter hinzuzufügen. Wenn Ihre HTML-Seite z. B. die `AC_FL_RunContent()`-Funktion verwendet (die in von Flash Professional und Flash Builder generierten HTML-Seiten verwendet wird), müssen Sie den `allowFullScreenInteractive`-Parameter wie folgt zum Funktionsaufruf hinzufügen:

```
AC_FL_RunContent(
    ...
    'allowFullScreenInteractive','true',
    ...
    ); //end AC code
```

Dies gilt nicht für SWF-Dateien, die im eigenständigen Flash Player ausgeführt werden.

Vollbild-Bühnengröße und Skalierung

Die Eigenschaften `Stage.fullScreenHeight` und `Stage.fullScreenWidth` geben die Höhe und die Breite des Monitors zurück, der für den Vollbildmodus verwendet wird, sofern dieser Status sofort aufgerufen wird. Diese Werte können falsch sein, wenn der Benutzer die Möglichkeit hat, den Browser von einem Monitor zu einem anderen zu bewegen, nachdem Sie diese Werte abgerufen haben, aber bevor Sie in den Vollbildmodus gewechselt sind. Wenn Sie diese Werte in derselben Ereignisprozedur abrufen, in der Sie die `Stage.displayState`-Eigenschaft auf `StageDisplayState.FULL_SCREEN` eingestellt haben, sind diese Werte korrekt. Wenn der Benutzer über mehrere Monitore verfügt, wird ein Monitor mit dem SWF-Inhalt ausgefüllt . Flash Player und AIR ermitteln über eine Metrik, welcher Monitor den größten Teil der SWF-Datei anzeigt, und verwenden diesen Monitor dann für den Vollbildmodus. Die fullScreenHeight- und fullScreenWidth-Eigenschaften beziehen sich nur auf die Größe des Monitors, der für den Vollbildmodus verwendet wird. Weitere Informationen finden Sie unter `Stage.fullScreenHeight` und `Stage.fullScreenWidth` im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Das Skalierungsverhalten der Bühne im Vollbildmodus entspricht dem im normalen Modus; die Skalierung wird durch die Eigenschaft `scaleMode` der Stage-Klasse geregelt. Wenn die `scaleMode`-Eigenschaft auf `StageScaleMode.NO_SCALE` eingestellt wird, ändern sich die Eigenschaften `stageWidth` und `stageHeight` der Bühne und geben die Größe des Bildschirmbereichs an, der von der SWF-Datei belegt wird (in diesem Fall der gesamte Bildschirm); bei Anzeige im Browser wird die Einstellung vom HTML-Parameter für diesen gesteuert.

Sie können das `fullScreen`-Ereignis der Stage-Klasse verwenden, um das Aktivieren oder Deaktivieren des Vollbildmodus zu erkennen und darauf zu reagieren. Beispielsweise möchten Sie beim Aktivieren oder Deaktivieren des Vollbildmodus Objekte auf dem Bildschirm neu positionieren, hinzufügen oder entfernen. Dies wird im folgenden Beispiel gezeigt:

```
import flash.events.FullScreenEvent;

function fullScreenRedraw(event:FullScreenEvent):void
{
    if (event.fullScreen)
    {
        // Remove input text fields.
        // Add a button that closes full-screen mode.
    }
    else
    {
        // Re-add input text fields.
        // Remove the button that closes full-screen mode.
    }
}

mySprite.stage.addEventListener(FullScreenEvent.FULL_SCREEN, fullScreenRedraw);
```

Wie dieser Code zeigt, ist das Ereignisobjekt für das `fullScreen`-Ereignis eine Instanz der flash.events.FullScreenEvent-Klasse, die eine `fullScreen`-Eigenschaft enthält, mit der angegeben wird, ob der Vollbildmodus aktiviert (`true`) oder deaktiviert (`false`) ist.

Tastaturunterstützung im Vollbildmodus

Wenn Flash Player in einem Browser ausgeführt wird, ist im Vollbildmodus der gesamte tastaturbezogene ActionScript-Code, zum Beispiel Tastaturereignisse und Texteingaben in TextField-Instanzen, deaktiviert. Es gibt folgende Ausnahmen (Tasten, die aktiviert sind):

- Bestimmte Tasten für nichtdruckbare Zeichen, speziell die Pfeiltasten, die Leertaste und die Tabulatortaste
- Tastaturbefehle, die den Vollbildmodus beenden: Esc (Windows und Mac), Strg-W (Windows), Befehl-W (Mac) und Alt-F4

Diese Einschränkungen gelten nicht für SWF-Inhalt, der im eigenständigen Flash Player oder in AIR ausgeführt wird. AIR unterstützt einen interaktiven Vollbildmodus, der Tastatureingaben zulässt.

Mausunterstützung im Vollbildmodus

Standardmäßig funktionieren Mausereignisse im Vollbildmodus genauso wie im Fenstermodus. Im Vollbildmodus können Sie jedoch optional die `Stage.mouseLock`-Eigenschaft festlegen, um die Maussperre zu aktivieren. Bei einer Maussperre ist der Cursor deaktiviert und unbegrenzte Mausbewegungen sind möglich.

Hinweis: *Sie können die Maussperre nur für Desktopanwendungen im Vollbildmodus aktivieren. Wenn Sie sie für Anwendungen, die nicht im Vollbildmodus ausgeführt werden, oder für Anwendungen auf mobilen Geräten festlegen, wird eine Ausnahme ausgegeben.*

Unter den folgenden Bedingungen wird die Maussperre automatisch deaktiviert und der Mauscursor ist wieder sichtbar:

- Der Benutzer beendet den Vollbildmodus, indem er die Escape-Taste drückt (alle Plattformen) oder eine der folgenden Tastenkombinationen verwendet: Strg+W (Windows), Befehl+W (Mac) oder Alt-F4 (Windows).
- Das Anwendungsfenster verliert den Fokus.
- Ein Element der Einstellungsbenutzeroberfläche ist sichtbar, zum Beispiel ein Datenschutzdialogfeld.
- Ein natives Dialogfeld wird angezeigt, zum Beispiel beim Hochladen einer Datei.

Ereignisse, die mit Mausbewegung verknüpft sind, zum Beispiel das `mouseMove`-Ereignis, verwenden die MouseEvent-Klasse, um das Ereignisobjekt darzustellen. Wenn die Maussperre deaktiviert ist, verwenden Sie die `MouseEvent.localX`- und die `MouseEvent.localY`-Eigenschaft, um die Position der Maus zu bestimmen. Wenn die Maussperre aktiviert ist, verwenden Sie die `MouseEvent.movementX`- und die `MouseEvent.movementY`-Eigenschaft, um die Position der Maus zu bestimmen. Die Eigenschaften `movementX` und `movementY` enthalten Änderungen an der Position der Maus seit dem letzten Ereignis anstelle absoluter Koordinaten für die Mausposition.

Hardwareskalierung im Vollbildmodus

Mit der `fullScreenSourceRect`-Eigenschaft der Stage-Klasse können Sie Flash Player oder AIR so konfigurieren, dass ein bestimmter Bereich der Bühne in den Vollbildmodus skaliert wird. Flash Player und AIR skalieren mit der Hardware, falls verfügbar, und verwenden die Grafikkarte des Benutzercomputers, wodurch Inhalte normalerweise schneller als mit Softwareskalierung angezeigt.

Um die Vorteile der Hardwareskalierung zu nutzen, setzen Sie die gesamte Bühne oder einen Teil der Bühne in den Vollbildmodus. Im folgenden ActionScript 3.0-Code wird die gesamte Bühne in den Vollbildmodus gesetzt:

```
import flash.geom.*;
{
    stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

Wenn diese Eigenschaft auf ein gültiges Rechteck und die Eigenschaft `displayState` auf den Vollbildmodus gesetzt wird, skalieren Flash Player und AIR den angegebenen Bereich. Die tatsächliche Größe der Bühne in Pixel innerhalb von ActionScript wird nicht geändert. Flash Player und AIR erzwingen eine Mindestgröße des Rechtecks, damit die standardmäßige Meldung „Vollbildmodus mit Esc beenden“ darin Platz findet. Diese Mindestgröße beträgt normalerweise ca. 260 x 30 Pixel, kann jedoch je nach Plattform und Flash Player-Version variieren.

Die `fullScreenSourceRect`-Eigenschaft kann nur festgelegt werden, wenn sich Flash Player bzw. AIR nicht im Vollbildmodus befindet. Um diese Eigenschaft korrekt zu verwenden, stellen Sie diese Eigenschaft zuerst ein und stellen Sie dann die `displayState`-Eigenschaft auf den Vollbildmodus ein.

Um die Skalierung zu aktivieren, legen Sie die Eigenschaft `fullScreenSourceRect` auf ein Rechteckobjekt fest.

```
stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
```

Zum Deaktivieren der Skalierung setzen Sie die `fullScreenSourceRect`-Eigenschaft auf `null`.

```
stage.fullScreenSourceRect = null;
```

Um alle Hardwarebeschleunigungsfunktionen mit Flash Player zu nutzen, aktivieren Sie sie über das Dialogfeld „Flash Player-Einstellungen“. Dieses Dialogfeld laden Sie, indem Sie mit der rechten Maustaste (Windows) oder bei gedrückter Ctrl-Taste (Mac) im Browser auf Flash Player-Inhalt klicken. Wählen Sie die Registerkarte „Anzeige“ und aktivieren Sie das Kontrollkästchen „Hardwarebeschleunigung aktivieren“.

Direkt- und GPU-Compositing-Fenstermodi

Flash Player 10 führt zwei Fenstermodi ein, „Direkt“ und „GPU-Compositing“, die Sie mithilfe der Veröffentlichungseinstellungen im Flash-Authoring-Tool aktivieren können. Diese Modi werden in AIR nicht unterstützt. Um diese Modi zu nutzen, müssen Sie die Hardwarebeschleunigung für Flash Player aktivieren.

Im Direktmodus werden Grafiken auf dem schnellsten, direktesten Pfad auf den Bildschirm gebracht, was bei der Videowiedergabe vorteilhaft ist.

Im GPU-Compositing-Modus wird das Compositing mithilfe der Graphikverarbeitungseinheit (Graphics Processing Unit, GPU) auf der Grafikkarte beschleunigt. Bei Video-Compositing handelt es sich um das Verfahren, bei dem mehrere Bilder übereinandergeschichtet werden, um ein Videobild zu erstellen. Durch die Beschleunigung des Compositing mithilfe der GPU kann die Leistung bei der YUV- Konvertierung, Farbkorrektur, Drehung, Skalierung und Mischung verbessert werden. YUV-Konvertierung verweist auf die Farbkonvertierung von zusammengesetzten, bei der Übertragung verwendeten Analogsignalen in das RGB-Farbmodell (rot, grün, blau), das von Videokameras und Bildschirmen verwendet wird. Durch die Beschleunigung des Compositing mithilfe der GPU werden die Speicher- und Rechenanforderungen verringert, die andernfalls an die CPU gestellt würden. Außerdem führt sie zu einer glatteren Wiedergabe von standardmäßigen Videos.

Gehen Sie bei der Implementierung dieser Fenstermodi vorsichtig vor. Das GPU-Compositing kann hohe Ansprüche an die Speicher- und CPU-Ressourcen stellen. Wenn Vorgänge (wie z. B. Mischmodi, Filtern, Zuschneiden oder Maskieren) nicht in der GPU ausgeführt werden können, müssen sie von der Software ausgeführt werden. Adobe empfiehlt, sich bei Verwendung dieser Modi auf eine SWF-Datei pro HTML-Seite zu beschränken, und rät davon ab, diese Modi für Banner zu aktivieren. Die Funktion „Film testen“ von Flash setzt keine Hardwarebeschleunigung ein, Sie können Sie aber über die Option „Vorschau für Veröffentlichungen“ verwenden.

Wenn Sie in der SWF-Datei eine Bildrate von über 60 einstellen, ist die maximale Aktualisierungsrate des Bildschirms nutzlos. Bei einer Bildrate zwischen 50 und 55 werden übersprungene Bilder in Betracht gezogen, die hin und wieder aus verschiedenen Gründen auftreten können.

Für den Direktmodus ist Microsoft DirectX 9 mit 128 MB VRAM unter Windows bzw. OpenGL unter Apple Macintosh, Mac OS X v10.2 oder höher erforderlich. Für das GPU-Compositing ist Microsoft DirectX 9 und Pixel Shader 2.0-Unterstützung unter Windows mit 128 MB VRAM erforderlich. Unter Mac OS X und Linux sind für das GPU-Compositing OpenGL 1.5 sowie mehrere OpenGL-Erweiterungen erforderlich (Framebuffer-Objekt, Multitexture, Shader-Objekte, Shading Language, Fragment-Shader).

Sie können die Beschleunigungsmodi `direct` und `gpu` für einzelne SWF-Dateien aktivieren. Klicken Sie dazu im Dialogfeld „Einstellungen für Veröffentlichungen“ auf die Registerkarte „Flash“ und rufen Sie hier das Menü „Hardwarebeschleunigung“ auf. Wenn Sie „Keine“ auswählen, wird der Fenstermodus auf die entsprechende Einstellung in der Registerkarte „HTML“ zurückgesetzt, d. h. `default`, `transparent` oder `opaque`.

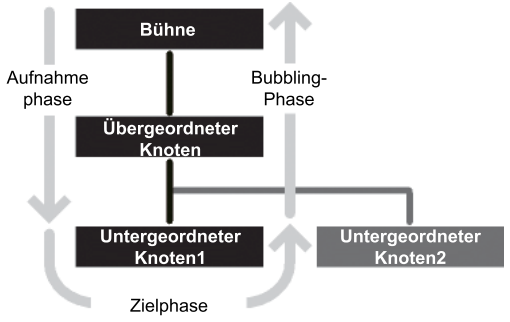
Verarbeiten von Ereignissen bei Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die DisplayObject-Klasse erbt von der EventDispatcher-Klasse. Dies bedeutet, dass jedes Anzeigeobjekt vollständig am Ereignismodell teilnehmen kann (siehe Beschreibung in „Verarbeiten von Ereignissen“ auf Seite 133). Jedes Ereignisobjekt kann seine `addEventListener()`-Methode verwenden (geerbt von der EventDispatcher-Klasse), um auf ein bestimmtes Ereignis zu überwachen, aber nur dann, wenn das überwachende Objekt Teil des Ereignisablaufs für dieses Ereignis ist.

Wenn Flash Player oder AIR ein Ereignisobjekt auslöst, tritt es einen Weg von der Bühne bis zum Ereignisziel (dem Ereignisobjekt, an dem das Ereignis aufgetreten ist) und zurück an. Klickt ein Benutzer beispielsweise auf ein Anzeigeobjekt namens `Untergeordneter Knoten 1`, sendet Flash Player ein Ereignisobjekt von der Bühne durch die Hierarchie der Anzeigeliste bis hinunter zum Anzeigeobjekt `Untergeordneter Knoten 1`.

Der Ereignisablauf ist, wie im folgenden Diagramm dargestellt, im Prinzip in drei Phasen unterteilt:



Weitere Informationen finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Ein wichtiger Punkt muss beim Arbeiten mit Anzeigeobjektereignissen berücksichtigt werden. Ereignis-Listener können beeinflussen, ob Anzeigeobjekte nach dem Entfernen aus der Anzeigeliste automatisch vom Speichermanagement (Garbage Collection) gelöscht werden. Wenn ein Anzeigeobjekt Objekte hat, die als Listener auf dessen Ereignissen abonniert sind, wird dieses Anzeigeobjekt auch dann nicht aus dem Speicher gelöscht, wenn es aus der Anzeigeliste entfernt wird, da noch immer Verweise auf diese Listener-Objekte vorhanden sind. Weitere Informationen finden Sie unter „Verwalten von Ereignis-Listenern“ auf Seite 148.

Auswählen einer DisplayObject-Unterklasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine der wichtigsten Entscheidungen, die Sie beim Arbeiten mit Anzeigeobjekten treffen müssen, ist, welches Anzeigeobjekt für welchen Zweck verwendet werden soll. Dabei stehen Ihnen verschiedene Optionen zur Auswahl. Im Folgenden finden Sie einige Richtlinien, die Ihnen bei der Entscheidung helfen sollen. Die gleichen Vorschläge gelten, wenn Sie eine Instanz einer Klasse benötigen oder eine Basisklasse für eine von Ihnen erstellte Klasse wählen müssen:

- Wenn Sie kein Objekt als Container für andere Anzeigeobjekte benötigen (d. h. Sie benötigen ein Objekt als alleinstehendes Bildschirmelement), wählen Sie eine der folgenden DisplayObject- oder InteractiveObject-Unterklassen, je nachdem, wofür sie benötigt wird:
  - Bitmap zur Anzeige eines Bitmapbilds.
  - TextField zum Hinzufügen von Text.
  - Video zur Anzeige von Video.
  - Shape als eine „Leinwand“ für das Zeichnen von Inhalten auf dem Bildschirm. Wenn Sie eine Instanz zum Zeichnen von Formen auf dem Bildschirm erstellen möchten, die kein Container für andere Bildschirmobjekte ist, erzielen Sie durch das Verwenden von Shape anstelle von Sprite oder MovieClip eine deutliche Leistungssteigerung.
  - MorphShape, StaticText oder SimpleButton für Objekte, die mit dem Flash-Authoring-Tool erstellt wurden. (Sie können keine programmgesteuerten Instanzen dieser Klassen erstellen, aber Sie können Variablen mit diesen Datentypen erstellen, die auf Objekte verweisen, die mit dem Flash-Authoring-Tool erstellt wurden.)
- Wenn Sie eine Variable benötigen, die auf die Hauptbühne verweist, verwenden Sie die Stage-Klasse als ihren Datentyp.

- Wenn Sie einen Container zum Laden einer externen SWF-Datei oder einer Grafikdatei benötigen, verwenden Sie eine Loader-Instanz. Der geladene Inhalt wird der Anzeigeliste als untergeordnetes Element der Loader-Instanz hinzugefügt. Der Datentyp des untergeordneten Elements hängt vom geladenen Inhalt ab. Dabei gilt Folgendes:
  - Ein geladenes Bild ist eine Bitmap-Instanz.
  - Eine geladene SWF-Datei, die in ActionScript 3.0 geschrieben wurde, ist eine Sprite- oder MovieClip-Instanz (oder eine Instanz einer Unterklasse dieser Klassen, wenn dies durch den Inhaltsersteller vorgegeben ist).
  - Eine geladene SWF-Datei, die in ActionScript 1.0 oder ActionScript 2.0 geschrieben wurde, ist eine AVM1Movie-Instanz.
- Wenn Sie ein Objekt benötigen, das als Container für andere Anzeigeobjekte dient (unabhängig davon, ob Sie mit ActionScript auf das Anzeigeobjekt zeichnen), wählen Sie eine der DisplayObjectContainer-Unterklassen:
  - Sprite, wenn das Objekt nur mit ActionScript erstellt wird oder als Basisklasse für ein benutzerdefiniertes Anzeigeobjekt dient, das ausschließlich mit ActionScript erstellt und bearbeitet wird.
  - MovieClip, wenn Sie eine Variable erstellen, die auf ein Movieclip-Symbol verweist, das in der Flash-Authoring-Umgebung erstellt wurde.
- Wenn Sie eine Klasse erstellen, die einem Movieclip-Symbol in der Flash Library zugeordnet wird, wählen Sie eine der folgenden DisplayObjectContainer-Unterklassen als Basisklasse Ihrer Klasse:
  - MovieClip, wenn das zugewiesene Movieclip-Symbol in mindestens einem Bild Inhalt aufweist.
  - Sprite, wenn das zugewiesene Movieclip-Symbol nur im ersten Bild Inhalt aufweist.

Bearbeiten von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Unabhängig davon, welches Anzeigeobjekt Sie verwenden, gibt es eine Reihe von Manipulationen, die alle Anzeigeobjekte gemeinsam haben, da sie Elemente sind, die auf dem Bildschirm angezeigt werden. Beispielsweise können alle Objekte auf dem Bildschirm positioniert, im Stapel nach oben bzw. nach unten verschoben, skaliert, gedreht usw. werden. Da alle Anzeigeobjekte diese Funktionsmerkmale von ihrer gemeinsamen Basisklasse erben (DisplayObject), verhält sich ein Funktionsmerkmal immer gleich, unabhängig davon, ob Sie eine TextField-Instanz, eine Video-Instanz, eine Shape-Instanz oder ein anderes Anzeigeobjekt bearbeiten. In den folgenden Abschnitten erfahren Sie mehr zu den Manipulationsmöglichkeiten von allgemeinen Anzeigeobjekten.

Ändern der Position

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine allgemeine Manipulation eines Anzeigeobjekts ist das Positionieren des Objekts auf dem Bildschirm. Zum Einstellen der Position eines Anzeigeobjekts ändern Sie dessen Eigenschaften `x` und `y`.

```
myShape.x = 17;
myShape.y = 212;
```

Im Positionierungssystem von Anzeigeobjekten wird die Bühne als kartesisches Koordinatensystem (das gewohnte Rastersystem mit einer horizontalen x- und einer vertikalen y-Achse) behandelt. Der Ursprung des Koordinatensystems (die Koordinate 0,0, an der sich die x- und y-Achse treffen), befindet sich in der oberen linken Ecke der Bühne. Von diesem Ursprungspunkt aus gehen die positiven x-Werte nach rechts und die negativen nach links, während (im Gegensatz zu typischen Graphensystemen) positive y-Werte nach unten und negative nach oben gehen. Beispielsweise wird das Objekt `myShape` mit den folgenden Codezeilen an die x-Koordinate 17 (17 Pixel vom Ursprungspunkt nach rechts) und die y-Koordinate 212 (212 Pixel vom Ursprungspunkt nach unten) verschoben.

Standardmäßig werden die `x`- und `y`-Eigenschaften eines mit ActionScript erstellten Anzeigeobjekts auf 0 eingestellt und das Objekt somit in der oberen linken Ecke des übergeordneten Inhalts platziert.

Ändern der Position relativ zur Bühne

Die Eigenschaften `x` und `y` beziehen sich immer auf die Position des Anzeigeobjekts relativ zur 0,0-Koordinate der Achsen des übergeordneten Anzeigeobjekts. Bei einer Shape-Instanz (z. B. einem Kreis), die in einer Sprite-Instanz enthalten ist, platziert das Einstellen der `x`- und `y`-Eigenschaften auf Null den Kreis in der oberen linken Ecke des Sprite, die nicht unbedingt auch die obere linke Ecke der Bühne ist. Um ein Objekt relativ zu den globalen Koordinaten der Bühne zu platzieren, verwenden Sie die `globalToLocal()`-Methode eines Anzeigeobjekts, um die Koordinaten von globalen Koordinaten (Bühne) in lokale Koordinaten (Anzeigeobjektcontainer) umzuwandeln. Dies wird im folgenden Beispiel gezeigt:

```
// Position the shape at the top-left corner of the Stage,
// regardless of where its parent is located.

// Create a Sprite, positioned at x:200 and y:200.
var mySprite:Sprite = new Sprite();
mySprite.x = 200;
mySprite.y = 200;
this.addChild(mySprite);

// Draw a dot at the Sprite's 0,0 coordinate, for reference.
mySprite.graphics.lineStyle(1, 0x000000);
mySprite.graphics.beginFill(0x000000);
mySprite.graphics.moveTo(0, 0);
mySprite.graphics.lineTo(1, 0);
mySprite.graphics.lineTo(1, 1);
mySprite.graphics.lineTo(0, 1);
mySprite.graphics.endFill();

// Create the circle Shape instance.
var circle:Shape = new Shape();
mySprite.addChild(circle);

// Draw a circle with radius 50 and center point at x:50, y:50 in the Shape.
circle.graphics.lineStyle(1, 0x000000);
circle.graphics.beginFill(0xff0000);
circle.graphics.drawCircle(50, 50, 50);
circle.graphics.endFill();

// Move the Shape so its top-left corner is at the Stage's 0, 0 coordinate.
var stagePoint:Point = new Point(0, 0);
var targetPoint:Point = mySprite.globalToLocal(stagePoint);
circle.x = targetPoint.x;
circle.y = targetPoint.y;
```

Entsprechend können Sie die `localToGlobal()`-Methode der DisplayObject-Klasse verwenden, um lokale Koordinaten in globale Bühnenkoordinaten umzuwandeln.

Verschieben von Anzeigeobjekten mit der Maus

Mithilfe von zwei verschiedenen ActionScript-Techniken können Sie Benutzern die Möglichkeit geben, Anzeigeobjekte mit der Maus zu verschieben. In beiden Fällen werden zwei Mausereignisse verwendet: Beim Drücken der Maustaste wird das Objekt angewiesen, dem Mauszeiger zu folgen; beim Loslassen der Maustaste wird das Objekt angewiesen, dem Mauszeiger nicht weiter zu folgen.

Hinweis: *Ab Flash Player 11.3 und ab AIR 3.3: Sie können auch das MouseEvent.RELEASE_OUTSIDE-Ereignis verwenden, um den Fall abzudecken, wenn ein Benutzer die Maustaste außerhalb der Grenzen des umhüllenden Sprites loslässt.*

Das erste Verfahren, bei dem die `startDrag()`-Methode verwendet wird, ist einfacher, aber auch weniger vielseitig. Beim Drücken der Maustaste wird die `startDrag()`-Methode des zu ziehenden Anzeigeobjekts aufgerufen. Beim Loslassen der Maustaste wird die `stopDrag()`-Methode aufgerufen. Diese beiden Funktionen werden von der Sprite-Klasse definiert. Deshalb muss das verschobene Objekt zur Sprite-Klasse oder einer ihrer Unterklassen gehören.

```
// This code creates a mouse drag interaction using the startDrag()
// technique.
// square is a MovieClip or Sprite instance).

import flash.events.MouseEvent;

// This function is called when the mouse button is pressed.
function startDragging(event:MouseEvent):void
{
    square.startDrag();
}

// This function is called when the mouse button is released.
function stopDragging(event:MouseEvent):void
{
    square.stopDrag();
}

square.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
square.addEventListener(MouseEvent.MOUSE_UP, stopDragging);
```

Dieses Verfahren weist jedoch eine entscheidende Einschränkung auf: Mit `startDrag()` kann nur jeweils ein Objekt gezogen werden. Wenn ein Anzeigeobjekt gezogen wird und die `startDrag()`-Methode für ein weiteres Anzeigeobjekt aufgerufen wird, hört das erste Anzeigeobjekt sofort auf, dem Mauszeiger zu folgen. Nach einer Änderung der `startDragging()`-Funktion (wie im Folgenden gezeigt), wird nur das `circle`-Objekt gezogen, ungeachtet des Aufrufs der `square.startDrag()`-Methode:

```
function startDragging(event:MouseEvent):void
{
    square.startDrag();
    circle.startDrag();
}
```

Als Konsequenz der Tatsache, dass mit `startDrag()` nur ein Objekt gezogen werden kann, soll die `stopDrag()`-Methode für ein Anzeigeobjekt aufgerufen werden, um das derzeit gezogene Objekt zu stoppen.

Wenn Sie mehrere Anzeigeobjekte ziehen müssen oder mögliche Konflikte vermeiden möchten, wenn mehrere Objekte `startDrag()` verwenden, verwenden Sie am besten die Maus folgen-Technik, um den Zieheffekt zu erzeugen. Bei dieser Technik wird beim Drücken der Maustaste eine Funktion als Listener des `mouseMove`-Ereignisses der Bühne abonniert. Diese Funktion, die mit jeder Bewegung der Maus aufgerufen wird, führt dazu, dass das gezogene Objekt zur x,y-Koordinate der Maus springt. Sobald die Maustaste losgelassen wird, ist die Funktion nicht mehr als Listener abonniert. Dies bedeutet, dass sie nicht mehr mit jeder Bewegung der Maus aufgerufen wird und das Objekt dem Mauszeiger nicht mehr folgt. Im Folgenden ist ein Code aufgeführt, der diese Technik demonstriert:

```
// This code moves display objects using the mouse-following
// technique.
// circle is a DisplayObject (e.g. a MovieClip or Sprite instance).

import flash.events.MouseEvent;

var offsetX:Number;
var offsetY:Number;

// This function is called when the mouse button is pressed.
function startDragging(event:MouseEvent):void
{
    // Record the difference (offset) between where
    // the cursor was when the mouse button was pressed and the x, y
    // coordinate of the circle when the mouse button was pressed.
    offsetX = event.stageX - circle.x;
    offsetY = event.stageY - circle.y;

    // tell Flash Player to start listening for the mouseMove event
    stage.addEventListener(MouseEvent.MOUSE_MOVE, dragCircle);
}

// This function is called when the mouse button is released.
function stopDragging(event:MouseEvent):void
{
    // Tell Flash Player to stop listening for the mouseMove event.
    stage.removeEventListener(MouseEvent.MOUSE_MOVE, dragCircle);
}

// This function is called every time the mouse moves,
// as long as the mouse button is pressed down.
function dragCircle(event:MouseEvent):void
{
    // Move the circle to the location of the cursor, maintaining
    // the offset between the cursor's location and the
    // location of the dragged object.
    circle.x = event.stageX - offsetX;
    circle.y = event.stageY - offsetY;

    // Instruct Flash Player to refresh the screen after this event.
    event.updateAfterEvent();
}

circle.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
circle.addEventListener(MouseEvent.MOUSE_UP, stopDragging);
```

Neben dem Effekt, dass ein Anzeigeobjekt dem Mauszeiger folgt, ist es oft empfehlenswert, das gezogene Objekt in den Vordergrund der Anzeige zu stellen, sodass es schwebend vor allen anderen Objekten angezeigt wird. Angenommen, Sie haben zwei Objekte, die beide mit der Maus verschoben werden können: einen Kreis und ein Quadrat. Wenn sich der Kreis in der Anzeigeliste unter dem Quadrat befindet und Sie den Kreis klicken und ziehen, sodass sich der Mauszeiger über dem Quadrat befindet, scheint sich der Kreis hinter das Quadrat zu schieben, wodurch der Ziehen-und-Ablegen-Eindruck unterbrochen wird. Stattdessen können Sie es so einrichten, dass der Kreis nach dem Klicken an die Spitze der Anzeigeliste verschoben wird und somit immer im Vordergrund vor allen anderen Objekten angezeigt wird.

Mit dem folgenden Code (der aus dem vorigen Beispiel übernommen und angepasst wurde) können zwei Anzeigeobjekte, ein Kreis und ein Quadrat, mit der Maus verschoben werden. Wenn die Maustaste über einem dieser Objekte gedrückt wird, wandert dieses Objekt an die Spitze der Anzeigeliste der Bühne, sodass es während des Ziehens immer im Vordergrund angezeigt wird. (Im Vergleich zum vorangegangenen Codebeispiel neuer oder geänderter Code wird fett angezeigt.)

```
// This code creates a drag-and-drop interaction using the mouse-following
// technique.
// circle and square are DisplayObjects (e.g. MovieClip or Sprite
// instances).

import flash.display.DisplayObject;
import flash.events.MouseEvent;

var offsetX:Number;
var offsetY:Number;
var draggedObject:DisplayObject;

// This function is called when the mouse button is pressed.
function startDragging(event:MouseEvent):void
{
    // remember which object is being dragged
    draggedObject = DisplayObject(event.target);

    // Record the difference (offset) between where the cursor was when
    // the mouse button was pressed and the x, y coordinate of the
    // dragged object when the mouse button was pressed.
    offsetX = event.stageX - draggedObject.x;
    offsetY = event.stageY - draggedObject.y;

    // move the selected object to the top of the display list
    stage.addChild(draggedObject);

    // Tell Flash Player to start listening for the mouseMove event.
    stage.addEventListener(MouseEvent.MOUSE_MOVE, dragObject);
}

// This function is called when the mouse button is released.
function stopDragging(event:MouseEvent):void
{
    // Tell Flash Player to stop listening for the mouseMove event.
    stage.removeEventListener(MouseEvent.MOUSE_MOVE, dragObject);
}
```

```
// This function is called every time the mouse moves,
// as long as the mouse button is pressed down.
function dragObject(event:MouseEvent):void
{
    // Move the dragged object to the location of the cursor, maintaining
    // the offset between the cursor's location and the location
    // of the dragged object.
    draggedObject.x = event.stageX - offsetX;
    draggedObject.y = event.stageY - offsetY;

    // Instruct Flash Player to refresh the screen after this event.
    event.updateAfterEvent();
}

circle.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
circle.addEventListener(MouseEvent.MOUSE_UP, stopDragging);

square.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
square.addEventListener(MouseEvent.MOUSE_UP, stopDragging);
```

Um diesen Effekt zu erweitern, beispielsweise für ein Spiel, bei dem Token oder Karten zwischen Stapeln verschoben werden, können Sie das gezogene Objekt beim „Aufnehmen“ zur Anzeigeliste der Bühne hinzufügen und es dann beim „Ablegen“, d. h. dem Loslassen der Maustaste, einer anderen Anzeigeliste hinzufügen (z. B. der Anzeigeliste „Stapel“).

Als weitere Aufwertung können Sie einen Drop Shadow-Filter auf ein Anzeigeobjekt anwenden, wenn darauf geklickt wird (wenn Sie das Ziehen beginnen), und diesen Schlagschatten mit dem Loslassen des Objekts wieder entfernen. Weitere Informationen zum Drop Shadow-Filter und anderen Filtern für Anzeigeobjekte in ActionScript finden Sie unter „Anwenden von Filtern auf Anzeigeobjekte“ auf Seite 286.

Schwenken und Scrollen von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie ein Anzeigeobjekt haben, das zu groß für den Bereich ist, in dem es angezeigt werden soll, können Sie den sichtbaren Bereich des Anzeigeobjekts mit der Eigenschaft `scrollRect` definieren. Darüber hinaus können Sie durch Ändern der `scrollRect`-Eigenschaft als Reaktion auf eine Benutzereingabe den Inhalt von links nach rechts schwenken oder nach oben und unten scrollen lassen.

Die `scrollRect`-Eigenschaft ist eine Instanz der Rectangle-Klasse. Diese Klasse fasst die zum Definieren eines rechteckigen Bereichs als einzelnes Objekt erforderlichen Werte zusammen. Um zunächst einmal den sichtbaren Bereich des Anzeigeobjekts zu definieren, erstellen Sie eine neue Rectangle-Instanz und weisen sie der Eigenschaft `scrollRect` des Anzeigeobjekts zu. Später, zum Scrollen oder Schwenken, lesen Sie die Eigenschaft `scrollRect` in eine separate Rectangle-Variable und ändern die gewünschte Eigenschaft (zum Schwenken die Eigenschaft `x` des Rechtecks und zum Scrollen die Eigenschaft `y)`. Dann weisen Sie dieser Rectangle-Instanz die `scrollRect`-Eigenschaft erneut zu, um das Anzeigeobjekt über den geänderten Werte zu informieren.

Im folgenden Beispielcode wird der sichtbare Bereich eines TextField-Objekts namens `bigText` definiert, das zu hoch für die Begrenzungen der SWF-Datei ist. Durch Klicken auf die zwei Schaltflächen `up` und `down` werden Funktionen aufgerufen, die den Inhalt des TextField-Objekts durch Ändern der `y`-Eigenschaft der Rectangle-Instanz `scrollRect` nach oben oder unten scrollen.

```
import flash.events.MouseEvent;
import flash.geom.Rectangle;

// Define the initial viewable area of the TextField instance:
// left: 0, top: 0, width: TextField's width, height: 350 pixels.
bigText.scrollRect = new Rectangle(0, 0, bigText.width, 350);

// Cache the TextField as a bitmap to improve performance.
bigText.cacheAsBitmap = true;

// called when the "up" button is clicked
function scrollUp(event:MouseEvent):void
{
    // Get access to the current scroll rectangle.
    var rect:Rectangle = bigText.scrollRect;
    // Decrease the y value of the rectangle by 20, effectively
    // shifting the rectangle down by 20 pixels.
    rect.y -= 20;
    // Reassign the rectangle to the TextField to "apply" the change.
    bigText.scrollRect = rect;
}

// called when the "down" button is clicked
function scrollDown(event:MouseEvent):void
{
    // Get access to the current scroll rectangle.
    var rect:Rectangle = bigText.scrollRect;
    // Increase the y value of the rectangle by 20, effectively
    // shifting the rectangle up by 20 pixels.
    rect.y += 20;
    // Reassign the rectangle to the TextField to "apply" the change.
    bigText.scrollRect = rect;
}

up.addEventListener(MouseEvent.CLICK, scrollUp);
down.addEventListener(MouseEvent.CLICK, scrollDown);
```

Dieses Beispiel zeigt, dass beim Arbeiten mit der `scrollRect`-Eigenschaft eines Anzeigeobjekts Flash Player angewiesen werden sollte, den Inhalt des Anzeigeobjekts mit der `cacheAsBitmap`-Eigenschaft als Bitmap zwischenzuspeichern. Dann müssen Flash Player und AIR den gesamten Inhalt beim Scrollen des Anzeigeobjekts nicht jedes Mal neu zeichnen. Stattdessen kann die zwischengespeicherte Bitmap verwendet werden, um den erforderlichen Teil direkt auf dem Bildschirm darzustellen. Nähere Einzelheiten finden Sie unter „Zwischenspeichern von Anzeigeobjekten“ auf Seite 194.

Ändern der Größe und Skalieren von Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die Größe eines Anzeigeobjekts auf zwei Arten messen und ändern. Dabei verwenden Sie entweder die Abmessungseigenschaften (`width` und `height`) oder die Skalierungseigenschaften (`scaleX` und `scaleY`).

Jedes Anzeigeobjekt hat eine `width`- und eine `height`-Eigenschaft, welche die ursprüngliche Größe des Objekts in Pixel festlegen. Sie können die Werte dieser Eigenschaften auslesen, um die Größe des Anzeigeobjekts zu messen. Außerdem können Sie neue Werte angeben, um die Größe des Objekts zu ändern. Dies wird im folgenden Beispiel gezeigt:

```
// Resize a display object.
square.width = 420;
square.height = 420;

// Determine the radius of a circle display object.
var radius:Number = circle.width / 2;
```

Das Ändern von `height` oder `width` eines Anzeigeobjekts führt zu einer Skalierung des Objekts; dies bedeutet, dass der Inhalt entweder gestreckt oder gestaucht wird, damit er in den neuen Bereich passt. Wenn das Anzeigeobjekt nur Vektorformen enthält, werden diese ohne Qualitätsverlust in der neuen Skalierung neu gezeichnet. Bitmap-Elemente im Anzeigeobjekt werden eher verkleinert als neu gezeichnet. Bei einem Digitalfoto, dessen Breite und Höhe über die tatsächlichen Abmessungen des Anzeigeobjekts hinausreichen, werden die Pixelinformationen des Bilds geändert, und es könnte gezackt aussehen oder es sind einzelne Pixelpunkte zu sehen.

Wenn Sie die Eigenschaften `width` oder `height` eines Anzeigeobjekts ändern, aktualisieren Flash Player und AIR auch die Eigenschaften `scaleX` und `scaleY` des Objekts.

Hinweis: *TextField-Objekte bilden bei diesem Skalierungsverhalten eine Ausnahme. Textfelder müssen ihre Größe selbst an Textumbrüche und Schriftgrößen anpassen. Deshalb setzen sie ihre Eigenschaften „scaleX“ oder „scaleY“ nach der Größenänderung auf den Wert 1 zurück. Wenn Sie jedoch die Werte „scaleX“ oder „scaleY“ eines TextField-Objekts anpassen, ändern sich die Werte für Breite und Höhe (width, height) gemäß den von Ihnen angegebenen Skalierungswerten.*

Diese Eigenschaften repräsentieren die relative Größe des Anzeigeobjekts im Vergleich zur Originalgröße. Die Eigenschaften `scaleX` und `scaleY` verwenden Brüche (Dezimalwerte) zur Darstellung von Prozentzahlen. Angenommen, die Eigenschaft `width` eines Anzeigeobjekts wurde geändert, sodass sie nur noch die Hälfte der ursprünglichen Größe beträgt. In diesem Fall nimmt die Eigenschaft `scaleX` des Objekts den Wert `.5` an, also 50 Prozent. Wurde die Höhe verdoppelt, nimmt die Eigenschaft `scaleY` den Wert `2` an, also 200 Prozent.

```
// circle is a display object whose width and height are 150 pixels.
// At original size, scaleX and scaleY are 1 (100%).
trace(circle.scaleX); // output: 1
trace(circle.scaleY); // output: 1

// When you change the width and height properties,
// Flash Player changes the scaleX and scaleY properties accordingly.
circle.width = 100;
circle.height = 75;
trace(circle.scaleX); // output: 0.6622516556291391
trace(circle.scaleY); // output: 0.4966887417218543
```

Größenänderungen erfolgen nicht proportional. Anders ausgedrückt, wenn Sie die Eigenschaft `height` eines Quadrats ändern, jedoch nicht die Eigenschaft `width`, so sind die Proportionen nicht mehr gleich, und die Form gleicht einem Rechteck anstatt einem Quadrat. Wenn Sie relative Änderungen an der Größe eines Anzeigeobjekts vornehmen möchten, können Sie alternativ zum Einstellen der Eigenschaften `width` oder `height` die Werte der Eigenschaften `scaleX` und `scaleY` ändern. Beispielsweise ändert der folgende Code die Eigenschaft `width` des Anzeigeobjekts `square` und ändert dann den vertikalen Maßstab (`scaleY`) so, dass er dem horizontalen Maßstab entspricht und die Größe des Quadrats proportional bleibt.

```
// Change the width directly.
square.width = 150;

// Change the vertical scale to match the horizontal scale,
// to keep the size proportional.
square.scaleY = square.scaleX;
```

Steuern der Verzerrung beim Skalieren

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Normalerweise wird die beim Skalieren eines Anzeigeobjekts resultierende Verzerrung (beispielsweise horizontales Strecken) gleichmäßig über das ganze Objekt verteilt, sodass jedes Teil um den gleichen Betrag gestreckt ist. Bei Grafiken und Designelementen wird dieses Verhalten gewünscht. Manchmal ist es jedoch von Vorteil, bestimmen zu können, welche Teile eines Anzeigeobjekts gestreckt werden und welche Teile unverändert bleiben sollen. Ein allgemeines Beispiel hierfür ist eine rechteckige Schaltfläche mit gerundeten Ecken. Bei einer normalen Skalierung werden auch die Ecken der Schaltfläche gestreckt, wodurch sich der Eckenradius ändert, wenn die Größe der Schaltfläche geändert wird.

In diesem Fall möchten Sie wahrscheinlich die Skalierung steuern können – um Bereiche festzulegen, die skaliert werden sollen (die geraden Seiten und die Mitte), und Bereiche, die nicht skaliert werden sollen (die Ecken) – sodass die Skalierung ohne sichtbare Verzerrung erfolgt.

Mit der Skalierung im 9-teiligen Segmentraster (Scale-9) können Sie Anzeigeobjekte erstellen, bei denen Sie festlegen können, wie sie skaliert werden. Bei der Skalierung im 9-teiligen Segmentraster wird das Anzeigeobjekt in neun separate Rechtecke aufgeteilt (ein 3x3 Raster, wie das Raster eines Tic-Tac-Toe-Spiels.) Die Rechtecke haben nicht unbedingt die gleiche Größe – Sie können festlegen, wo die Rasterlinien platziert werden. Jeder Inhalt, der sich in den vier Eck-Rechtecken befinden (beispielsweise die gerundeten Ecken einer Schaltfläche), wird bei einer Skalierung des Anzeigeobjekts weder gestreckt noch gestaucht. Das obere mittlere und untere mittlere Rechteck werden horizontal, aber nicht vertikal skaliert, während das linke mittlere und das rechte mittlere Rechteck vertikal, aber nicht horizontal skaliert werden. Das zentrale Rechteck wird sowohl horizontal als auch vertikal skaliert.

Wenn Sie also ein Anzeigeobjekt erstellen und sicherstellen möchten, dass bestimmte Inhalte nicht skaliert werden, achten Sie darauf, dass die Teilungslinien des 9-teiligen Segmentrasters so platziert sind, dass der Inhalt in einem der Eck-Rechtecke endet.

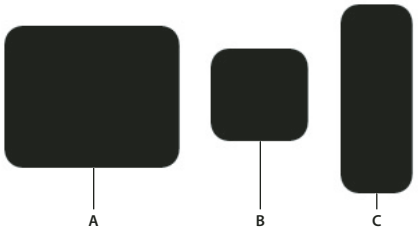
In ActionScript aktiviert das Einstellen eines Werts für die Eigenschaft `scale9Grid` eines Anzeigeobjekts das 9-teilige Segmentraster und legt die Größe der Rechtecke im Scale-9-Raster des Objekts fest. Sie können eine Instanz der Rectangle-Klasse als Wert für die Eigenschaft `scale9Grid` verwenden. Dies wird im folgenden Beispiel gezeigt:

```
myButton.scale9Grid = new Rectangle(32, 27, 71, 64);
```

Die vier Parameter des Rectangle-Konstruktors sind x-Koordinate, y-Koordinate, Breite und Höhe. In diesem Beispiel wird die obere linke Ecke des Rechtecks am Punkt x: 32, y: 27 auf dem Anzeigeobjekt `myButton` platziert. Das Rechteck ist 71 Pixel breit und 64 Pixel hoch (also befindet sich die rechte Ecke an der x-Koordinate 103 auf dem Anzeigeobjekt und die untere Ecke an der y-Koordinate 92 auf dem Anzeigeobjekt).

Der Bereich der von der Rectangle-Instanz definierten Fläche ist das zentrale Rechteck des Scale-9-Rasters. Die anderen Rechtecke werden von Flash Player und AIR berechnet, indem die Seiten der Rectangle-Instanz wie im Folgenden gezeigt erweitert werden:

In diesem Fall werden die gerundeten Ecken beim Vergrößern oder Verkleinern der Schaltflächen weder gestreckt noch gestaucht, die anderen Bereiche aber werden an die Skalierung angepasst.



A. *myButton.width = 131;myButton.height = 106;* ***B.*** *myButton.width = 73;myButton.height = 69;* ***C.*** *myButton.width = 54;myButton.height = 141;*

Zwischenspeichern von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Ihre Entwürfe in Flash umfangreicher werden, z. B. wenn Sie eine Anwendung oder komplexe Animationsskripts entwickeln, müssen Sie Leistung und Optimierung berücksichtigen. Bei statischen Inhalten (wie rechteckigen Shape-Instanzen) nehmen Flash Player und AIR keine Optimierung des Inhalts vor. Wenn Sie die Position des Rechtecks ändern, würde die gesamte Shape-Instanz in Flash Player oder AIR neu gezeichnet werden.

Sie können bestimmte Anzeigeobjekte zwischenspeichern, um die Leistung Ihrer SWF-Datei zu verbessern. Das Anzeigeobjekt kann mit einer *Oberfläche* verglichen werden, die im Prinzip eine Bitmap-Version der Vektordaten der Instanz darstellt. Hierbei handelt es sich um die Daten in der SWF-Datei, die sich voraussichtlich nur wenig ändern werden. Deshalb werden Instanzen, für die das Zwischenspeichern aktiviert ist, beim Abspielen der SWF-Datei nicht kontinuierlich neu gezeichnet, sodass die SWF-Datei schneller gerendert werden kann.

Hinweis: *Sie können die Vektordaten aktualisieren, wodurch die Oberfläche neu erstellt wird. Deshalb müssen die in der Oberfläche zwischengespeicherten Vektordaten nicht in der gesamten SWF-Datei unverändert bleiben.*

Durch das Einstellen der `cacheAsBitmap`-Eigenschaft eines Anzeigeobjekts auf `true` legt der Cache-Speicher des Anzeigeobjekts eine Bitmap-Darstellung von sich selbst an. Flash Player oder AIR erstellt ein Oberflächenobjekt der Instanz, bei dem es sich um eine zwischengespeicherte Bitmap anstelle von Vektordaten handelt. Wenn Sie die Abmessungen des Anzeigeobjekts ändern, wird die Oberfläche nicht vergrößert oder verkleinert, sondern neu erstellt. Oberflächen können innerhalb von anderen Oberflächen verschachtelt sein. Die untergeordnete Oberfläche kopiert die Bitmap auf die übergeordnete Oberfläche. Weitere Informationen finden Sie unter „Aktivieren der Bitmap-Zwischenspeicherung“ auf Seite 197.

Die Eigenschaften `opaqueBackground` und `scrollRect` der DisplayObject-Klasse sind mit der Bitmap-Zwischenspeicherung mit der `cacheAsBitmap`-Eigenschaft verwandt. Obwohl diese drei Eigenschaften unabhängig voneinander sind, arbeiten die Eigenschaften `opaqueBackground` und `scrollRect` am besten, wenn ein Objekt als eine Bitmap zwischengespeichert ist – Sie sehen die Leistungsvorteile für die Eigenschaften `opaqueBackground` und `scrollRect` nur dann, wenn Sie `cacheAsBitmap` auf `true` einstellen. Weitere Informationen zum Scrollen des Inhalts von Anzeigeobjekten finden Sie unter „Schwenken und Scrollen von Anzeigeobjekten“ auf Seite 190. Weitere Informationen zum Festlegen eines undurchsichtigen Hintergrunds finden Sie unter „Festlegen eines undurchsichtigen Hintergrunds“ auf Seite 197.

Weitere Informationen zur Maskierung des Alphakanals, für die Sie die Eigenschaft `cacheAsBitmap` auf `true` einstellen müssen, finden Sie unter „Maskieren von Anzeigeobjekten“ auf Seite 202.

Geeignete Szenarien für das Zwischenspeichern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie das Zwischenspeichern für ein Anzeigeobjekt aktivieren, wird eine Oberfläche erstellt. Dies bietet unter anderem den Vorteil, dass komplexe Vektoranimationen schneller gerendert werden. Das Zwischenspeichern ist in mehreren Szenarien eine gute Lösung, führt jedoch nicht immer zu einer Leistungssteigerung der SWF-Dateien, sondern kann in manchen Fällen die Leistung sogar beeinträchtigen. In diesem Abschnitt wird beschrieben, in welchen Szenarien das Zwischenspeichern zu empfehlen ist und wann Sie reguläre Anzeigeobjekte verwenden sollten.

Die Gesamtleistung der zwischengespeicherten Daten richtet sich nach den folgenden Faktoren: Komplexität der Vektordaten in den Instanzen, Menge der zu ändernden Daten und Einstellung der Eigenschaft `opaqueBackground`. Wenn Sie nur kleine Bereiche ändern, ist der Unterschied bei Verwendung einer Oberfläche im Vergleich mit Vektordaten möglicherweise kaum wahrnehmbar. Es empfiehlt sich, beide Szenarien zu testen, bevor Sie die Anwendung bereitstellen.

Szenarien für die Bitmap-Zwischenspeicherung

Im Folgenden werden einige typische Szenarien aufgelistet, bei denen die Bitmap-Zwischenspeicherung deutliche Vorteile bieten kann.

- Komplexe Hintergrundbilder: Eine Anwendung, die ein detailliertes und komplexes Hintergrundbild von Vektordaten enthält (beispielsweise ein Bild, auf das Sie den Befehl „Bitmap nachzeichnen“ angewendet haben, oder eine Grafik, die Sie in Adobe Illustrator® erstellt haben). Sie können Zeichen über dem Hintergrund animieren; dies verlangsamt die Animation, da der Hintergrund die Vektordaten laufend neu erstellen muss. Zur Leistungssteigerung können Sie die Eigenschaft `opaqueBackground` des Hintergrund-Anzeigeobjekts auf `true` einstellen. Der Hintergrund wird als Bitmap dargestellt und kann schnell neu gezeichnet werden, sodass die Animation schneller abgespielt wird.
- Scrollbares Textfeld: Eine Anwendung, die viel Text in einem scrollbaren Textfeld anzeigt. Sie können das Textfeld in einem Anzeigeobjekt platzieren, das Sie als scrollbar mit scrollbaren Begrenzungen festlegen (die Eigenschaft `scrollRect)`. Dies aktiviert schnelles Pixelscrolling für diese Instanz. Wenn ein Benutzer die Anzeigeobjekt-Instanz scrollt, verlagert Flash Player oder AIR die gescrollten Pixel nach oben und erstellt den neuen, freien Bereich anstelle des gesamten Textfelds.

- Windowing-System: Eine Anwendung mit komplexen, überlappenden Fenstern. Jedes Fenster kann geöffnet oder geschlossen werden (z. B. Webbrowserfenster). Wenn Sie jedes Fenster als Oberfläche markieren (durch Einstellen der Eigenschaft `cacheAsBitmap` auf `true`), wird jedes Fenster isoliert und zwischengespeichert. Benutzer können die Fenster ziehen, sodass sie überlappen, und kein Fenster muss den Vektorinhalt neu erstellen.
- Maskierung des Alphakanals: Wenn Sie die Maskierung des Alphakanals verwenden, müssen Sie die Eigenschaft `cacheAsBitmap` auf `true` setzen. Weitere Informationen finden Sie unter „Maskieren von Anzeigeobjekten“ auf Seite 202.

Das Aktivieren der Bitmap-Zwischenspeicherung für alle diese Szenarien verbessert die Reaktion und Interaktivität der Anwendung, da die Vektorgrafiken optimiert werden.

Wenn Sie darüber hinaus einen Filter auf das Anzeigeobjekt anwenden, wird die Eigenschaft `cacheAsBitmap` automatisch auf `true` eingestellt, auch wenn Sie explizit den Wert `false` vorgeben. Wenn Sie sämtliche Filter eines Anzeigeobjekts löschen, wird die Eigenschaft `cacheAsBitmap` auf den Wert zurückgesetzt, der zuletzt eingestellt war.

Nicht geeignete Szenarien für die Bitmap-Zwischenspeicherung

Wenn Sie dieses Funktionsmerkmal in den falschen Situationen verwenden, kann dies die Leistung der SWF-Datei beeinträchtigen. Beachten Sie bei der Verwendung der Bitmap-Zwischenspeicherung folgende Richtlinien:

- Verwenden Sie nicht zu viele Oberflächen (Anzeigeobjekte, für die das Zwischenspeichern aktiviert ist). Da jede Oberfläche mehr Arbeitsspeicher belegt als ein reguläres Anzeigeobjekt, sollten Sie Oberflächen nur aktivieren, wenn Sie die Leistung beim Rendern steigern müssen.

  Eine zwischengespeicherte Bitmap kann wesentlich mehr Speicher als ein reguläres Anzeigeobjekt belegen. Angenommen, eine Sprite-Instanz auf der Bühne hat eine Größe von 250 x 250 Pixel, so belegt die zwischengespeicherte Sprite-Instanz 250 KB und die reguläre Sprite-Instanz 1 KB.
- Vermeiden Sie das Vergrößern von zwischengespeicherten Oberflächen. Wenn Sie die Bitmap-Zwischenspeicherung zu häufig einsetzen, wird viel Speicher verbraucht (siehe vorherigen Punkt), besonders wenn Sie den Inhalt vergrößern.
- Verwenden Sie Oberflächen für Anzeigeobjekt-Instanzen, die im Wesentlichen statisch sind (also keine Animationen enthalten). Sie können die Instanz ziehen oder verschieben, doch der Inhalt der Instanz sollte nicht animiert sein und keine großen Änderungen aufweisen. (Animationen oder sich ändernde Inhalte sind eher bei einer MovieClip-Instanz mit einer Animation oder einer Video-Instanz zu finden.) Wenn Sie beispielsweise eine Instanz drehen oder transformieren, wechselt die Instanz zwischen der Oberfläche und Vektordaten. Dies ist schwer zu verarbeiten und wirkt sich negativ auf die SWF-Datei aus.
- Bei einer Kombination aus Oberflächen und Vektordaten ist der Verarbeitungsaufwand für Flash Player und AIR (und manchmal auch für den Computer) höher. Gruppieren Sie Oberflächen so oft wie möglich – z. B. beim Erstellen von Windowing-Anwendungen.
- Sie sollten keine Objekte zwischenspeichern, deren Grafiken sich häufig ändern. Der Bitmap-Zwischenspeicher wird bei den folgenden Aktionen immer neu gezeichnet: Skalieren, Neigen oder Drehen des Anzeigeobjekts, Ändern der Alpha- oder Farbtransformation, Verschieben von untergeordneten Anzeigeobjekten oder Zeichnen mit der graphics-Eigenschaft. Wenn dies bei jedem Bild der Fall ist, muss die Laufzeit das Objekt in eine Bitmap zeichnen und dann diese Bitmap auf die Bühne kopieren. Dies bedeutet mehr Arbeit, als wenn das nicht zwischengespeicherte Objekt einfach auf der Bühne gezeichnet wird. Ein angemessenes Leistungsverhältnis zwischen der Zwischenspeicherung und der Aktualisierungshäufigkeit richtet sich nach der Komplexität und Größe des Anzeigeobjekts. Letztendlich lässt sich dies nur durch Testen des jeweiligen Inhalts feststellen.

Aktivieren der Bitmap-Zwischenspeicherung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Aktivieren der Bitmap-Zwischenspeicherung für ein Anzeigeobjekt stellen Sie die Eigenschaft `cacheAsBitmap` auf `true` ein:

```
mySprite.cacheAsBitmap = true;
```

Nachdem Sie die Eigenschaft `cacheAsBitmap` auf `true` eingestellt haben, werden Sie eventuell feststellen, dass die Pixel im Anzeigeobjekt automatisch an ganzen Koordinaten ausgerichtet werden. Beim Testen der SWF-Datei werden Sie wahrscheinlich sehen, dass komplexe Vektoranimationen im Vergleich zu normalen Animationen wesentlich schneller gerendert werden.

In den folgenden Fällen wird keine Oberfläche (zwischengespeicherte Bitmap) erstellt, auch dann nicht, wenn `cacheAsBitmap` auf `true` eingestellt ist:

- Höhe oder Breite der Bitmap betragen mehr als 2880 Pixel.
- Der Bitmap kann (aufgrund von zu wenig Systemspeicher) nicht genügend Speicher zugewiesen werden.

Transformationsmatrizen für zwischengespeicherte Bitmaps

Adobe AIR 2.0 und höher (Mobilprofil)

In AIR-Anwendungen für Mobilgeräte sollten Sie die `cacheAsBitmapMatrix`-Eigenschaft immer dann festlegen, wenn Sie auch die `cacheAsBitmap`-Eigenschaft setzen. Durch Festlegen dieser Eigenschaft können Sie mehr Transformationen auf das Anzeigeobjekt anwenden, ohne dass ein erneutes Rendern ausgelöst wird.

```
mySprite.cacheAsBitmap = true;
mySprite.cacheAsBitmapMatrix = new Matrix();
```

Wenn Sie diese Matrixeigenschaft festlegen, können Sie die folgenden zusätzlichen Transformationen auf das Anzeigeobjekt anwenden, ohne dass das Objekt erneut zwischengespeichert wird:

- Verschieben oder Versetzen ohne Pixelausrichtung
- Drehen
- Skalieren
- Neigen
- Ändern des Alphawertes (zwischen 0 und 100 % Transparenz)

Diese Transformationen werden direkt auf die zwischengespeicherte Bitmap angewendet.

Festlegen eines undurchsichtigen Hintergrunds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können einen undurchsichtigen Hintergrund für ein Anzeigeobjekt einstellen. Angenommen, Ihre SWF-Datei enthält einen Hintergrund mit einer komplexen Vektorgrafik, so können Sie die Eigenschaft`opaqueBackground` auf eine bestimmte Farbe (normalerweise dieselbe Farbe wie die Bühne) einstellen. Die Farbe wird als Zahl angegeben (in der Regel als hexadezimaler Farbwert). Der Hintergrund wird dann als Bitmap behandelt, wodurch die Leistung optimiert wird.

Wenn Sie `cacheAsBitmap` auf `true` und die Eigenschaft `opaqueBackground` auf eine bestimmte Farbe einstellen, wird die interne Bitmap aufgrund der `opaqueBackground`-Eigenschaft schneller undurchsichtig dargestellt. Wenn Sie `cacheAsBitmap` nicht auf `true` einstellen, fügt die Eigenschaft `opaqueBackground` dem Hintergrund des Anzeigeobjekts eine undurchsichtige, quadratische Vektorform hinzu. Eine Bitmap wird nicht automatisch erstellt.

Im folgenden Beispielcode wird dargestellt, wie Sie den Hintergrund eines Anzeigeobjekts einstellen, um die Leistung zu optimieren.

```
myShape.cacheAsBitmap = true;
myShape.opaqueBackground = 0xFF0000;
```

In diesem Fall wird die Hintergrundfarbe der Form `myShape` auf Rot (`0xFF0000`) eingestellt. Angenommen, die Shape-Instanz enthält die Zeichnung eines grünen Dreiecks auf einer Bühne mit weißem Hintergrund, so würde ein grünes Dreieck mit roter Farbe im leeren Raum des Begrenzungsrahmens der Shape-Instanz angezeigt (das Rechteck, das die Form vollständig umschließt).

Natürlich würde dieser Code mehr Sinn ergeben, wenn er für eine Bühne mit einem vollständig roten Hintergrund verwendet wird. Bei einem andersfarbigen Hintergrund würde stattdessen diese Farbe angegeben werden. Beispielsweise würde die Eigenschaft `opaqueBackground` bei einer SWF-Datei mit weißem Hintergrund wahrscheinlich auf `0xFFFFFF`, reines Weiß, eingestellt werden.

Anwenden von Mischmodi

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei den Mischmodi werden die Farben eines Bilds (Basisbild) mit denen eines anderen Bilds (Mischbild) kombiniert, um ein drittes Bild zu erstellen. Das resultierende Bild wird schließlich auf dem Bildschirm angezeigt. Jeder Pixelwert in einem Bild wird mit dem entsprechenden Pixelwert des anderen Bilds verarbeitet. Daraus ergibt sich ein bestimmter Pixelwert an der betreffenden Position im dritten Bild.

Jedes Anzeigeobjekt weist die Eigenschaft `blendMode` auf, die auf einen der folgenden Mischmodi gesetzt werden kann. Hierbei handelt es sich um Konstanten, die in der BlendMode-Klasse definiert sind. Alternativ können Sie die String-Werte (in runden Klammern) verwenden, bei denen es sich um die tatsächlichen Werte der Konstanten handelt.

- `BlendMode.ADD` („`add`"): Erstellt animierte, heller werdende Auflösungen zwischen zwei Bildern.
- `BlendMode.ALPHA` („`alpha`"): Wendet die Transparenz des Vordergrunds auf den Hintergrund an. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.DARKEN` („`darken`"): Wird für Überlagerungen verwendet. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.DIFFERENCE` („`difference`"): Wird zur Verstärkung der Farben verwendet.
- `BlendMode.ERASE` („`erase`"): Dient zum Ausschneiden (Löschen) eines Teils des Hintergrunds mithilfe des Alphawerts des Vordergrunds. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.HARDLIGHT` („`hardlight`"): Dient zum Erstellen von Schatteneffekten. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.INVERT` („`invert`"): Wird zum Umkehren des Hintergrunds verwendet.

- `BlendMode.LAYER` („`layer`"): Erzwingt die Erstellung eines temporären Speichers für die vorläufige Komposition eines bestimmten Anzeigeobjekts. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.LIGHTEN` („`lighten`"): Wird für Überlagerungen verwendet. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.MULTIPLY` („`multiply`"): Dient zum Erstellen von Schatten- und Tiefeneffekten.
- `BlendMode.NORMAL` („`normal`"): Gibt an, dass die Pixelwerte des Mischbilds die des Basisbilds außer Kraft setzen.
- `BlendMode.OVERLAY` („`overlay`"): Dient zum Erstellen von Schatteneffekten. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.SCREEN` („`screen`"): Dient zum Erstellen von Hervorhebungs- und Linseneffekten.
- `BlendMode.SHADER` („`shader`"): Gibt an, dass ein Pixel Bender-Shader zum Erstellen eines benutzerdefinierten Mischeffekts verwendet werden soll. Weitere Informationen zur Verwendung von Shadern finden Sie unter „Arbeiten mit Pixel Bender-Shadern“ auf Seite 321. (Nicht unterstützt für GPU-Rendering.)
- `BlendMode.SUBTRACT` („`subtract`"): Erstellt animierte, dunkler werdende Auflösungen zwischen zwei Bildern.

Einstellen der DisplayObject-Farben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den Methoden der ColorTransform-Klasse (flash.geom.ColorTransform) können Sie die Farbe eines Anzeigeobjekts einstellen. Jedes Anzeigeobjekt hat eine `transform`-Eigenschaft, bei der es sich um eine Instanz der Transform-Klasse handelt. Sie enthält Informationen über verschiedene Transformationen, die auf das Anzeigeobjekt angewendet werden (z. B. Drehung, Änderung der Skalierung oder Position usw.). Neben den Informationen zu geometrischen Transformationen enthält die Transform-Klasse auch eine `colorTransform`-Eigenschaft, eine Instanz der ColorTransform-Klasse, die Zugriff zum Ändern der Farbe des Anzeigeobjekts bietet. Sie können beispielsweise den folgenden Code verwenden, um auf die Informationen der Farbtransformation eines Anzeigeobjekts zuzugreifen:

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;
```

Nachdem Sie eine ColorTransform-Instanz erstellt haben, können Sie deren Eigenschaftswerte auslesen, um die angewendeten Farbtransformationen anzuzeigen, oder Sie können die Werte einstellen, um Farbänderungen am Anzeigeobjekt vorzunehmen. Zum Aktualisieren des Anzeigeobjekts nach vorgenommenen Änderungen müssen Sie die ColorTransform-Instanz der Eigenschaft `transform.colorTransform` erneut zuweisen.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;

// Make some color transformations here.

// Commit the change.
myDisplayObject.transform.colorTransform = colorInfo;
```

Festlegen von Farbwerten im Programmcode

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Eigenschaft `color` der ColorTransform-Klasse können Sie dem Anzeigeobjekt einen bestimmten Rot-, Grün- oder Blauwert (RGB-Farbwert) zuweisen. Im folgenden Beispielcode wird mit der `color`-Eigenschaft die Farbe des Anzeigeobjekts `square` zu Blau geändert, wenn der Benutzer auf eine Schaltfläche mit der Bezeichnung `blueBtn` klickt:

```
// square is a display object on the Stage.
// blueBtn, redBtn, greenBtn, and blackBtn are buttons on the Stage.

import flash.events.MouseEvent;
import flash.geom.ColorTransform;

// Get access to the ColorTransform instance associated with square.
var colorInfo:ColorTransform = square.transform.colorTransform;

// This function is called when blueBtn is clicked.
function makeBlue(event:MouseEvent):void
{
    // Set the color of the ColorTransform object.
    colorInfo.color = 0x003399;
    // apply the change to the display object
    square.transform.colorTransform = colorInfo;
}

blueBtn.addEventListener(MouseEvent.CLICK, makeBlue);
```

Beachten Sie, dass beim Ändern der Farbe eines Anzeigeobjekts mit der Eigenschaft `color` die Farbe des gesamten Objekts vollständig geändert wird, unabhängig davon, ob das Objekt zuvor mehrfarbig war. Hierzu ein Beispiel: Wenn bei einem Anzeigeobjekt mit einem grünen Kreis und einem darauf angezeigten schwarzen Text die Eigenschaft `color` der diesem Objekt zugewiesenen ColorTransform-Instanz die Farbe Rot zugewiesen wird, so wird das gesamte Objekt einschließlich Kreis und Text rot (d. h. der Text kann nicht mehr vom übrigen Objekt unterschieden werden).

Ändern der Farb- und Helligkeitseffekte mit Code

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Angenommen, Sie haben ein Anzeigeobjekt mit mehreren Farben (beispielsweise ein Digitalfoto) und Sie möchten nicht das gesamte Objekt vollständig neu kolorieren, sondern nur die Farbe eines Anzeigeobjekts basierend auf den vorhandenen Farben einstellen. Für dieses Szenario enthält die ColorTransform-Klasse eine Reihe von Multiplikator- und Offset-Eigenschaften, mit denen Sie diese Einstellungen vornehmen können. Die Multiplikator-Eigenschaften `redMultiplier`, `greenMultiplier`, `blueMultiplier` und `alphaMultiplier` arbeiten wie Farbfotografie-Filter (oder farbige Sonnenbrillen) und verstärken oder schwächen bestimmte Farben im Anzeigeobjekt. Mit den Offset-Eigenschaften (`redOffset`, `greenOffset`, `blueOffset` und `alphaOffset`) können bestimmte Farben des Objekts verstärkt oder Mindestwerte für eine bestimmte Farbe vorgegeben werden.

Diese Multiplikator- und Offset-Eigenschaften sind mit den erweiterten Farbeinstellungen identisch, die für Movieclip-Symbole in der Flash-Authoring-Umgebung verfügbar werden, wenn Sie im Eigenschafteninspektor im Popupmenü „Farbe“ die Option „Erweitert“ auswählen.

Im folgenden Beispielcode werden eine JPEG-Grafik geladen und eine Farbtransformation angewendet, die den Rot- und Grünkanal verändert, während sich der Mauszeiger entlang der x- und y-Achse bewegt. Da keine Offset-Werte angegeben wurden, ist in diesem Fall der Farbwert jedes Farbkanals auf dem Bildschirm ein Prozentwert der Originalfarbe im Bild. Dies bedeutet, dass das stärkste Rot oder Grün, das in einem bestimmten Pixel angezeigt wird, der ursprüngliche Rot- oder Grünanteil in diesem Pixel ist.

```
import flash.display.Loader;
import flash.events.MouseEvent;
import flash.geom.Transform;
import flash.geom.ColorTransform;
import flash.net.URLRequest;

// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

// This function is called when the mouse moves over the loaded image.
function adjustColor(event:MouseEvent):void
{
    // Access the ColorTransform object for the Loader (containing the image)
    var colorTransformer:ColorTransform = loader.transform.colorTransform;

    // Set the red and green multipliers according to the mouse position.
    // The red value ranges from 0% (no red) when the cursor is at the left
    // to 100% red (normal image appearance) when the cursor is at the right.
    // The same applies to the green channel, except it's controlled by the
    // position of the mouse in the y axis.
    colorTransformer.redMultiplier = (loader.mouseX / loader.width) * 1;
    colorTransformer.greenMultiplier = (loader.mouseY / loader.height) * 1;

    // Apply the changes to the display object.
    loader.transform.colorTransform = colorTransformer;
}

loader.addEventListener(MouseEvent.MOUSE_MOVE, adjustColor);
```

Drehen von Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Anzeigeobjekte können mit der Eigenschaft `rotation` gedreht werden. Sie können diesen Wert auslesen, um festzustellen, ob ein Objekt gedreht wurde, oder Sie können diese Eigenschaft auf eine Zahl (in Grad) einstellen, um das Objekt zu drehen. Die eingegebene Zahl ist eine Angabe in Grad, um die das Objekt gedreht werden soll. Im folgenden Beispielcode wird das Objekt `square` um 45 Grad gedreht (1/8 einer vollständigen Umdrehung):

```
square.rotation = 45;
```

Alternativ können Sie ein Anzeigeobjekt mit einer Transformationsmatrix drehen. Weitere Informationen hierzu finden Sie unter „Verwenden von geometrischen Objekten“ auf Seite 224.

Ein- oder Ausblenden von Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die Transparenz eines Anzeigeobjekts steuern, um es teilweise (oder vollständig) transparent zu machen, oder die Transparenz so ändern, dass ein Objekt scheinbar ein- oder ausgeblendet wird. Die Eigenschaft `alpha` der DisplayObject-Klasse legt die Transparenz (genauer gesagt, die Opazität) eines Anzeigeobjekts fest. Die Eigenschaft `alpha` kann auf einen Wert zwischen 0 und 1 eingestellt werden; 0 ist dabei vollständig durchsichtig, 1 vollständig undurchsichtig. Mit dem folgenden Beispielcode wird das Objekt `myBall` teilweise (50 Prozent) durchsichtig, wenn mit der Maus darauf geklickt wird:

```
function fadeBall(event:MouseEvent):void
{
    myBall.alpha = .5;
}
myBall.addEventListener(MouseEvent.CLICK, fadeBall);
```

Sie können die Transparenz eines Anzeigeobjekts auch mithilfe der Farbeinstellungen ändern, die über die ColorTransform-Klasse verfügbar sind. Weitere Informationen finden Sie unter „Einstellen der DisplayObject-Farben“ auf Seite 199.

Maskieren von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit einem Anzeigeobjekt, das Sie als Maske verwenden, können Sie eine Öffnung erstellen, die den Blick auf ein zweites, darunter liegendes Anzeigeobjekt freigibt.

Definieren einer Maske

Um anzugeben, dass ein Anzeigeobjekt die Maske für ein zweites Anzeigeobjekt ist, richten Sie das Maskenobjekt als `mask`-Eigenschaft des zu maskierenden Anzeigeobjekts ein:

```
// Make the object maskSprite be a mask for the object mySprite.
mySprite.mask = maskSprite;
```

Das maskierte Anzeigeobjekt wird unter allen undurchsichtigen Bereichen des Anzeigeobjekts angezeigt, das als Maske dient. Mit dem folgenden Code werden eine Shape-Instanz erstellt, die ein rotes Rechteck mit einer Größe von 100 x 100 Pixel enthält, und eine Sprite-Instanz, die einen blauen Kreis mit einem Radius von 25 Pixel enthält. Der Kreis wird durch Klicken als Maske für das Quadrat ausgewählt, sodass der einzige Teil des Quadrats, der noch sichtbar ist, der Teil ist, der durch den festen Teil des Kreises abgedeckt wird. Mit anderen Worten, es ist nur ein roter Kreis sichtbar.

```
// This code assumes it's being run within a display object container
// such as a MovieClip or Sprite instance.

import flash.display.Shape;

// Draw a square and add it to the display list.
var square:Shape = new Shape();
square.graphics.lineStyle(1, 0x000000);
square.graphics.beginFill(0xff0000);
square.graphics.drawRect(0, 0, 100, 100);
square.graphics.endFill();
this.addChild(square);

// Draw a circle and add it to the display list.
var circle:Sprite = new Sprite();
circle.graphics.lineStyle(1, 0x000000);
circle.graphics.beginFill(0x0000ff);
circle.graphics.drawCircle(25, 25, 25);
circle.graphics.endFill();
this.addChild(circle);

function maskSquare(event:MouseEvent):void
{
    square.mask = circle;
    circle.removeEventListener(MouseEvent.CLICK, maskSquare);
}

circle.addEventListener(MouseEvent.CLICK, maskSquare);
```

Das als Maske verwendete Anzeigeobjekt kann gezogen, animiert und dynamisch in der Größe geändert werden und separate Formen innerhalb einer Maske verwenden. Das Masken-Anzeigeobjekt muss der Anzeigeliste nicht unbedingt hinzugefügt werden. Andererseits muss sich das Maskenobjekt in der Anzeigeliste befinden, wenn Sie das Maskenobjekt zusammen mit der Bühne skalieren möchten oder wenn eine Benutzerinteraktion mit der Maske möglich sein soll (z. B. vom Benutzer gesteuertes Ziehen und Ändern der Größe). Die tatsächliche z-Reihenfolge (von vorne nach hinten) der Anzeigeobjekte spielt keine Rolle, solange das Maskenobjekt der Anzeigeliste hinzugefügt ist. (Das Maskenobjekt erscheint nur als Maske auf dem Bildschirm.) Handelt es sich bei dem Maskenobjekt um eine MovieClip-Instanz mit mehreren Bildern, werden alle Bilder in der Zeitleiste wiedergegeben, genau so, als ob das Objekt nicht als Maske verwendet wird. Sie entfernen eine Maske, indem Sie die Eigenschaft `mask` auf `null` einstellen:

```
// remove the mask from mySprite
mySprite.mask = null;
```

Sie können eine Maske jedoch nicht auf eine andere Maske anwenden. Die `alpha`-Eigenschaft eines Masken-Anzeigeobjekts kann nicht eingestellt werden. Außerdem werden in einem Anzeigeobjekt, das als Maske verwendet wird, nur Füllungen dargestellt; Striche werden ignoriert.

AIR 2

Wenn ein maskiertes Anzeigeobjekt zwischengespeichert wird, indem die `cacheAsBitmap`- und `cacheAsBitmapMatrix`-Eigenschaften gesetzt werden, muss die Maske ein untergeordnetes Element des maskierten Anzeigeobjekts ein. Ist das maskierte Anzeigeobjekt ein untergeordnetes Element eines zwischengespeicherten Anzeigeobjektcontainers, müssen sowohl die Maske als auch das Anzeigeobjekt untergeordnete Elemente dieses Containers sein. Wenn das maskierte Objekt ein untergeordnetes Element von mehr als einem zwischengespeicherten Anzeigeobjektcontainer ist, muss die Maske dem zwischengespeicherten Container untergeordnet sein, der dem maskierten Objekt in der Anzeigeliste am nächsten ist.

Maskieren von Geräteschriftarten

Sie können mit einem Anzeigeobjekt einen Text maskieren, der in einer Geräteschriftart definiert ist. Wenn Sie eine Anzeigeobjekt-Maske zum Maskieren eines Textes verwenden, der in einer Geräteschriftart definiert ist, wird der rechteckige Begrenzungsrahmen der Maske als Maskenform verwendet. Das heißt, wenn Sie eine nicht rechteckige Anzeigeobjekt-Maske für Text in Geräteschriftarten erstellen, entspricht die Form der in der SWF-Datei angezeigten Maske nicht der eigentlichen Maskenform, sondern der Form des rechteckigen Begrenzungsrahmens.

Maskieren des Alphakanals

Die Alphakanal-Maskierung wird unterstützt, wenn die Maske und die maskierten Anzeigeobjekte die Bitmap-Zwischenspeicherung verwenden. Dies wird im folgenden Beispiel gezeigt:

```
// maskShape is a Shape instance which includes a gradient fill.
mySprite.cacheAsBitmap = true;
maskShape.cacheAsBitmap = true;
mySprite.mask = maskShape;
```

Eine Anwendung der Alphakanal-Maskierung wäre, einen Filter am Maskenobjekt anzuwenden, und zwar unabhängig von dem Filter, der auf das maskierte Anzeigeobjekt angewendet wurde.

Im folgenden Beispiel wird eine externe Bilddatei auf die Bühne geladen. Das Bild (oder genauer gesagt die Loader-Instanz, in die es geladen ist) ist das maskierte Anzeigeobjekt. Über das Bild wird ein Verlaufsoval gezeichnet (eine Form mit einem festen schwarzen Mittelpunkt, der zu den Rändern hin transparent wird); dies ist die Alphamaske. Für beide Anzeigeobjekte muss die Bitmap-Zwischenspeicherung aktiviert sein. Das Oval wird als eine Maske für das Bild eingestellt und dann zu einem ziehbaren Objekt gemacht.

```
// This code assumes it's being run within a display object container
// such as a MovieClip or Sprite instance.

import flash.display.GradientType;
import flash.display.Loader;
import flash.display.Sprite;
import flash.geom.Matrix;
import flash.net.URLRequest;

// Load an image and add it to the display list.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

// Create a Sprite.
var oval:Sprite = new Sprite();
// Draw a gradient oval.
var colors:Array = [0x000000, 0x000000];
var alphas:Array = [1, 0];
var ratios:Array = [0, 255];
var matrix:Matrix = new Matrix();
matrix.createGradientBox(200, 100, 0, -100, -50);
oval.graphics.beginGradientFill(GradientType.RADIAL,
                                colors,
                                alphas,
                                ratios,
                                matrix);
oval.graphics.drawEllipse(-100, -50, 200, 100);
oval.graphics.endFill();
// add the Sprite to the display list
this.addChild(oval);

// Set cacheAsBitmap = true for both display objects.
loader.cacheAsBitmap = true;
oval.cacheAsBitmap = true;
// Set the oval as the mask for the loader (and its child, the loaded image)
loader.mask = oval;

// Make the oval draggable.
oval.startDrag(true);
```

Animieren von Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine Animation ist der Prozess, ein Objekt zu bewegen, oder alternativ, eine Änderung über die Zeit zu erzeugen. Animationsskripten sind ein grundlegender Teil von Videospielen und werden häufig dazu verwendet, Anwendungen aufzuwerten und nützliche Hinweise zur Interaktion anzuzeigen.

Das grundlegende Konzept von Animationsskripten ist, dass eine Änderung stattfinden muss, und diese Änderung muss in einzelne Schritte über die Zeit eingeteilt werden. Mit einer allgemeinen Schleifenanweisung ist es leicht, etwas in ActionScript zu wiederholen. Jedoch muss eine Schleife alle Iterationen durchlaufen, bevor die Anzeige aktualisiert wird. Zum Erstellen von Animationsskripts müssen Sie ActionScript schreiben, das bestimmte Aktionen wiederholt ausführt und den Bildschirm nach jedem Durchlauf aktualisiert.

Angenommen, Sie möchten eine einfache Animation erstellen, z. B. den Weg eines Balls über den Bildschirm. ActionScript enthält einen einfachen Mechanismus, mit dem Sie den Zeitverlauf erfassen und den Bildschirm entsprechend aktualisieren können – anders ausgedrückt, Sie können Code schreiben, der den Ball jedes Mal um einen kleinen Betrag weiter bewegt, bis er sein Ziel erreicht. Nach jeder Bewegung wird der Bildschirm aktualisiert, wodurch die Bewegung des Balls über die Bühne für den Betrachter sichtbar wird.

Vom praktischen Standpunkt aus macht es Sinn, Animationsskripts mit der Bildrate einer SWF-Datei zu synchronisieren (d. h. eine Animationsänderung einzuleiten, wenn ein neues Bild angezeigt wird oder angezeigt werden würde), da dies festlegt, wie häufig Flash Player oder AIR den Bildschirm aktualisiert. Jedes Anzeigeobjekt hat ein `enterFrame`-Ereignis, das entsprechend der Bildrate der SWF-Datei ausgelöst wird – ein Ereignis pro Bild. Die meisten Entwickler, die Animationsskripts erstellen, sehen in dem `enterFrame`-Ereignis eine Möglichkeit, Aktionen zu erstellen, die sich mit der Zeit wiederholen. Sie können Code schreiben, der das `enterFrame`-Ereignis überwacht und den animierten Ball in jedem Bild um eine bestimmte Strecke bewegt. Wenn der Bildschirm dann aktualisiert wird (bei jedem Bild), wird der Ball an seiner neuen Position neu gezeichnet, wodurch eine Bewegung entsteht.

Hinweis: *Eine andere Möglichkeit, eine Aktion wiederholt auszuführen, ist die Timer-Klasse. Eine Timer-Instanz löst jedes Mal, wenn eine bestimmte Zeit verstrichen ist, eine Ereignisbenachrichtigung aus. Sie können Code schreiben, der eine Animation durch Verarbeiten des timer-Ereignisses der Timer-Klasse ausführt. Stellen Sie dazu das Zeitintervall sehr klein ein (einige Bruchteile einer Sekunde). Weitere Informationen zur Verwendung der Timer-Klasse finden Sie unter „Steuern von Zeitintervallen“ auf Seite 4.*

Im folgenden Beispielcode wird eine kreisförmige Sprite-Instanz namens `circle` auf der Bühne erstellt. Klickt der Benutzer auf den Kreis, beginnt eine Animationsskriptsequenz, die `circle` ausblendet (die Eigenschaft `alpha` wird verringert), bis der Kreis vollständig transparent ist:

```
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;

// draw a circle and add it to the display list
var circle:Sprite = new Sprite();
circle.graphics.beginFill(0x990000);
circle.graphics.drawCircle(50, 50, 50);
circle.graphics.endFill();
addChild(circle);

// When this animation starts, this function is called every frame.
// The change made by this function (updated to the screen every
// frame) is what causes the animation to occur.
function fadeCircle(event:Event):void
{
    circle.alpha -= .05;

    if (circle.alpha <= 0)
    {
        circle.removeEventListener(Event.ENTER_FRAME, fadeCircle);
    }
}

function startAnimation(event:MouseEvent):void
{
    circle.addEventListener(Event.ENTER_FRAME, fadeCircle);
}

circle.addEventListener(MouseEvent.CLICK, startAnimation);
```

Klickt der Benutzer auf den Kreis, wird die Funktion `fadeCircle()` als Listener des Ereignisses `enterFrame` abonniert. Dies bedeutet, sie wird in jedem Bild einmal aufgerufen. Diese Funktion blendet `circle` aus, indem sie die Eigenschaft `alpha` ändert. Der `alpha`-Wert des Kreises wird in jedem Bild um 0,05 (5 Prozent) verringert und der Bildschirm aktualisiert. Wenn der `alpha`-Wert schließlich 0 beträgt (`circle` ist vollständig transparent), wird die `fadeCircle()`-Funktion als Ereignis-Listener entfernt und die Animation wird beendet.

Der gleiche Code könnte auch zum Erstellen einer animierten Bewegung anstelle einer Ausblendung verwendet werden. Durch Einsetzen einer anderen Eigenschaft für `alpha` in der Funktion, die ein `enterFrame`-Ereignis-Listener ist, wird die Eigenschaft stattdessen animiert. Beispielsweise wird durch Ändern der Zeile

```
    circle.alpha -= .05;
```

zu

```
    circle.x += 5;
```

die Eigenschaft `x` animiert und der Kreis führt eine Bewegung über die Bühne nach rechts aus. Die Bedingung, die das Ausblenden beendet, könnte zum Beenden der Animation geändert werden (d. h. zum Beenden des Abonnements des Listeners `enterFrame`), wenn die gewünschte x-Koordinate erreicht ist.

Bühnenausrichtung

AIR 2.0 und höher

Auf Mobilgeräten wird die Benutzeroberfläche meist neu ausgerichtet, wenn der Benutzer das Gerät dreht, damit sie immer aufrecht angezeigt wird. Wenn Sie die automatische Ausrichtung in Ihrer Anwendung aktivieren, wird die Anzeige auf dem Gerät stets korrekt ausgerichtet. Sie müssen jedoch dafür sorgen, dass der Inhalt richtig dargestellt wird, wenn das Seitenverhältnis der Bühne sich ändert. Wenn Sie die automatische Ausrichtung deaktivieren, ist die Anzeige auf dem Gerät fixiert, es sei denn, Sie ändern die Ausrichtung manuell.

AIR-Anwendungen können auf verschiedenen Mobilgeräten und Betriebssystemen ausgeführt werden. Das zugrunde liegende Ausrichtungsverhalten kann bei verschiedenen Betriebssystemen – und sogar bei verschiedenen Geräten unter demselben Betriebssystem – variieren. Als einfache Entwicklungsstrategie, die bei allen Geräten und Betriebssystemen gut funktioniert, empfiehlt es sich, die automatische Ausrichtung zu aktivieren und auf `resize`-Ereignisse der Bühne zu warten, um festzustellen, wann das Anwendungslayout aktualisiert werden muss.

Wenn Ihre Anwendung entweder nur das Hochformat oder nur das Querformat unterstützt, können Sie stattdessen auch die automatische Ausrichtung deaktivieren und das unterstützte Seitenverhältnis im AIR-Anwendungsdeskriptor festlegen. Diese Strategie sorgt für ein einheitliches Verhalten und wählt die „beste“ Ausrichtung für das ausgewählte Seitenverhältnis aus. Wenn Sie zum Beispiel das Querformat angeben, eignet sich die gewählte Ausrichtung für Geräte mit einer herausschiebbaren Tastatur im Querformat.

Abrufen der aktuellen Bühnenausrichtung und des aktuellen Seitenverhältnisses

Die Ausrichtung wird relativ zur normalen Position des Geräts angegeben. Auf den meisten Geräten ist eine eindeutig aufrechte Position vorhanden. Diese Position gilt als *Standardausrichtung*. Die drei anderen möglichen Ausrichtungen sind dann: *nach links gedreht*, *nach rechts gedreht* und *auf den Kopf gedreht*. Die StageOrientation-Klasse definiert Stringkonstanten, die zum Einstellen oder Vergleichen von Ausrichtungswerten verwendet werden können.

Die Stage-Klasse definiert zwei Eigenschaften zum Angeben der Ausrichtung:

- Stage.deviceOrientation – gibt die physische Ausrichtung des Geräts relativ zur Standardposition an.

  Hinweis: *Die Ausrichtung des Geräts (deviceOrientation) ist nicht immer verfügbar, wenn die Anwendung gestartet wird oder wenn das Gerät auf einer flachen Oberfläche liegt. In diesen Fällen wird die Geräteausrichtung als unbekannt angegeben.*

- Stage.orientation – gibt die Ausrichtung der Bühne relativ zur Standardposition an. Wenn die automatische Ausrichtung aktiviert ist, dreht sich die Bühne in entgegengesetzter Richtung zum Gerät, damit sie aufrecht bleibt. Deshalb gibt die `orientation`-Eigenschaft für die rechte und die linke Position das Gegenteil der Werte an, die von der `deviceOrientation`-Eigenschaft angegeben werden. Wenn `deviceRotation` beispielsweise *nach rechts gedreht* angibt, meldet `orientation`*nach links gedreht*.

Das Seitenverhältnis der Bühne kann ganz einfach ermittelt werden, indem die aktuelle Breite der Bühne mit der aktuellen Höhe verglichen wird:

```
var aspect:String = this.stage.stageWidth >= this.stage.stageHeight ?
StageAspectRatio.LANDSCAPE : StageAspectRatio.PORTRAIT;
```

Automatische Ausrichtung

Wenn die automatische Ausrichtung aktiviert ist und der Benutzer das Gerät dreht, richtet das Betriebssystem die ganze Benutzeroberfläche neu aus, also auch die Taskleiste des Systems und Ihre Anwendung. Als Resultat ändert sich das Seitenverhältnis der Bühne von Hochformat zu Querformat oder von Querformat zu Hochformat. Bei einer Änderung des Seitenverhältnisses ändern sich auch die Abmessungen der Bühne.

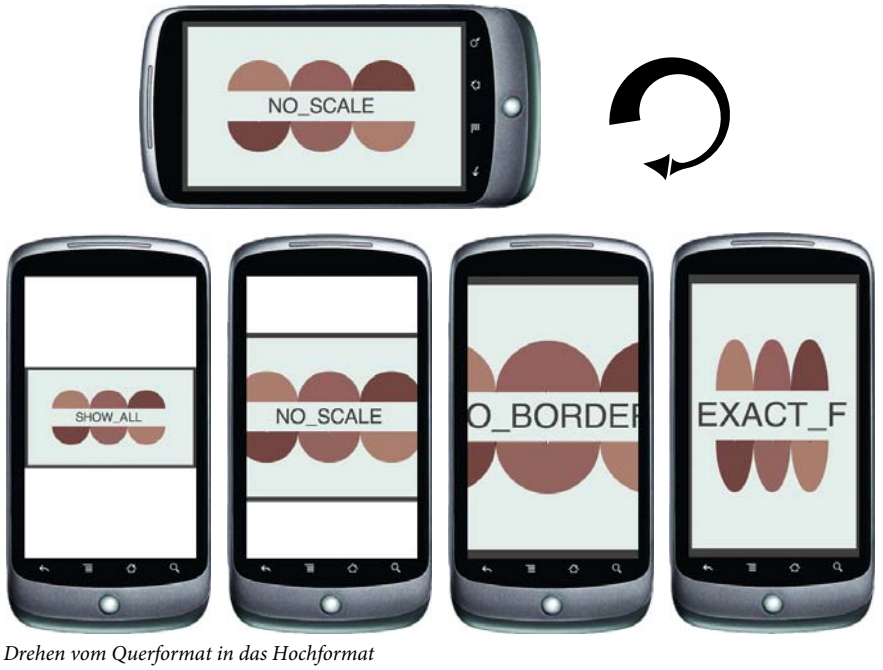
Sie können die automatische Ausrichtung zur Laufzeit aktivieren oder deaktivieren, indem Sie die `autoOrients`-Eigenschaft der Bühne auf `true` bzw. `false` setzen. Sie können den Anfangswert dieser Eigenschaft im AIR-Anwendungsdeskriptor über das `<autoOrients>`-Element festlegen. (In Versionen vor AIR 2.6 ist `autoOrients` eine schreibgeschützte Eigenschaft, die nur im Anwendungsdeskriptor festgelegt werden kann.)

Wenn Sie ein Querformat- oder Hochformat-Seitenverhältnis festlegen und auch die automatische Ausrichtung aktivieren, beschränkt AIR die automatische Ausrichtung auf das angegebene Seitenverhältnis.

Änderungen an den Abmessungen der Bühne

Bei einer Änderung der Bühnenabmessungen wird der Inhalt der Bühne skaliert und neu positioniert, wie von den `scaleMode`- und `align`-Eigenschaften des Stage-Objekts angegeben. In den meisten Fällen liefert das automatische Verhalten der `scaleMode`-Bühneneinstellungen keine guten Ergebnisse. Stattdessen sollten Sie Ihre Grafiken und Komponenten neu anordnen oder neu zeichnen, damit mehr als ein Seitenverhältnis unterstützt wird. (Eine flexible Layoutlogik bietet auch den Vorteil, dass Ihre Anwendung auf verschiedenen Geräten mit unterschiedlichen Bildschirmgrößen und Seitenverhältnissen besser funktioniert.)

Die folgenden Abbildungen zeigen die Auswirkungen verschiedener `scaleMode`-Einstellungen beim Drehen eines typischen Mobilgeräts:



Drehen vom Querformat in das Hochformat

Die Abbildung zeigt das Skalierungsverhalten beim Drehen vom Querformat in das Hochformat mit unterschiedlichen Skalierungsmodi. Beim Drehen vom Hochformat in das Querformat sind die Auswirkungen ähnlich.

Ereignisse bei der Ausrichtungsänderung

Das Stage-Objekt löst zwei verschiedene Ereignisarten aus, mit denen Sie Ausrichtungsänderungen erkennen und darauf reagieren können. Wenn die automatische Ausrichtung aktiviert ist, werden `resize`- und `orientationChange`-Ereignisse für die Bühne ausgelöst.

Das *resize*-Ereignis wird empfohlen, wenn Sie die automatische Ausrichtung verwenden, um die Anzeige in aufrechter Position zu halten. Wenn die Bühne ein `resize`-Ereignis auslöst, wird der Inhalt nach Bedarf neu angeordnet oder neu gezeichnet. Das `resize`-Ereignis wird nur ausgelöst, wenn der Skalierungsmodus der Bühne auf `noScale` eingestellt ist.

Das `orientationChange`-Ereignis kann auch verwendet werden, um Änderungen an der Ausrichtung zu erkennen. Das `orientationChange`-Ereignis wird nur ausgelöst, wenn die automatische Ausrichtung aktiviert ist.

Hinweis: *Auf einigen Mobilplattformen löst die Bühne ein abbrechbares `orientationChanging`-Ereignis aus, bevor das resize- oder das orientationChange-Ereignis ausgelöst wird. Da das Ereignis nicht auf allen Plattformen unterstützt wird, sollten Sie sich nach Möglichkeit nicht darauf verlassen.*

Manuelle Ausrichtung

AIR 2.6 und höher

Sie können die Bühnenausrichtung mit den `setOrientation()`- oder `setAspectRatio()`-Methoden steuern.

Einstellen der Bühnenausrichtung

Sie können die Bühnenausrichtung zur Laufzeit mit der `setOrientation()`-Methode des Stage-Objekts festlegen. Geben Sie die gewünschte Ausrichtung über die Stringkonstanten an, die von der StageOrientation-Klasse definiert werden:

```
this.stage.setOrientation( StageOrientation.ROTATED_RIGHT );
```

Nicht alle Geräte und Betriebssysteme unterstützen jede Ausrichtung, die theoretisch möglich ist. So unterstützt Android 2.2 beispielsweise nicht die programmatische Auswahl der nach links gedrehten Ausrichtung auf Geräten, die standardmäßig das Hochformat haben. Die Ausrichtung auf dem Kopf wird überhaupt nicht unterstützt. Die `supportedOrientations`-Eigenschaft der Bühne bietet eine Liste der Ausrichtungen, die an die `setOrientation()`-Methode übergeben werden können:

```
var orientations:Vector.<String> = this.stage.supportedOrientations;
for each( var orientation:String in orientations )
{
    trace( orientation );
}
```

Einstellen des Seitenverhältnisses der Bühne

Wenn das Seitenverhältnis der Bühne für Sie besonders wichtig ist, können Sie das Seitenverhältnis auf Hochformat oder Querformat einstellen. Das Seitenverhältnis kann entweder im AIR-Anwendungsdeskriptor oder zur Laufzeit über die `setAspectRatio()`-Methode der Bühne festgelegt werden:

```
this.stage.setAspectRatio( StageAspectRatio.LANDSCAPE );
```

Die Laufzeit wählt eine der beiden möglichen Ausrichtungen für das angegebene Seitenverhältnis. Diese Ausrichtung stimmt möglicherweise nicht mit der aktuellen Geräteausrichtung überein. Zum Beispiel wird die Standardausrichtung anstelle der auf dem Kopf stehenden Ausrichtung gewählt (AIR 3.2 und früher), und die Ausrichtung, die sich für die Slide-out-Tastatur eignet, wird anstelle der entgegengesetzten Ausrichtung gewählt.

(AIR 3.3 und höher) Beginnend mit AIR 3.3 (SWF-Version 16) können Sie auch die `StageAspectRatio.ANY`-Konstante verwenden. Wenn `Stage.autoOrients` auf `true` gesetzt ist und Sie `setAspectRatio(StageAspectRatio.ANY)` aufrufen, kann sich Ihre Anwendung in alle Ausrichtungen neu ausrichten (landscape-left, landscape-right, portrait, und portrait-upside-down). Ebenfalls neu in AIR 3.3 ist, dass das Seitenverhältnis persistent ist und die weitere Drehung des Geräts auf die angegebene Ausrichtung beschränkt ist.

Beispiel: Einstellen der Bühnenausrichtung entsprechend der Geräteausrichtung

Das folgende Beispiel zeigt eine Funktion, die die Bühnenausrichtung so aktualisiert, dass sie der aktuellen Geräteausrichtung entspricht. Die `deviceOrientation`-Eigenschaft der Bühne gibt die tatsächliche Ausrichtung des Geräts an, selbst wenn die automatische Ausrichtung deaktiviert ist.

```
function refreshOrientation( theStage:Stage ):void
{
    switch ( theStage.deviceOrientation )
    {
        case StageOrientation.DEFAULT:
            theStage.setOrientation( StageOrientation.DEFAULT );
            break;
        case StageOrientation.ROTATED_RIGHT:
            theStage.setOrientation( StageOrientation.ROTATED_LEFT );
            break;
        case StageOrientation.ROTATED_LEFT:
            theStage.setOrientation( StageOrientation.ROTATED_RIGHT );
            break;
        case StageOrientation.UPSIDE_DOWN:
            theStage.setOrientation( StageOrientation.UPSIDE_DOWN );
            break;
        default:
            //No change
    }
}
```

Die Änderung der Ausrichtung erfolgt asynchron. Sie können auf das von der Bühne ausgelöste `orientationChange`-Ereignis warten, um zu erkennen, wann die Änderung abgeschlossen ist. Wenn die Ausrichtung auf einem Gerät nicht unterstützt wird, schlägt der `setOrientation()`-Aufruf ohne Ausgabe einer Fehlermeldung fehl.

Dynamisches Laden von Anzeigeinhalten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können eines der folgenden externen Anzeigeelemente in eine ActionScript 3.0-Anwendung laden:

- Eine in ActionScript 3.0 erstellte SWF-Datei – diese Datei kann ein Sprite, MovieClip oder eine andere Klasse sein, die Sprite erweitert. In AIR-Anwendungen unter iOS können nur SWF-Dateien, die keinen ActionScript-Bytecode enthalten, geladen werden. SWF-Dateien mit eingebetteten Daten, wie Bilder und Sound, können also geladen werden, nicht aber SWF-Dateien, die ausführbaren Code enthalten.
- Eine Bilddatei – hierzu gehören JPG-, PNG- und GIF-Dateien.
- Eine AVM1 SWF-Datei – dies ist eine in ActionScript 1.0 oder 2.0 geschriebene SWF-Datei. (nicht in mobilen AIR-Anwendungen unterstützt)

Diese Anzeigeelemente können Sie mit der Loader-Klasse laden.

Laden von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Loader-Objekte dienen zum Laden von SWF- und Grafikdateien in eine Anwendung. Die Loader-Klasse ist eine Unterklasse der DisplayObjectContainer-Klasse. Ein Loader-Objekt kann nur ein untergeordnetes Anzeigeobjekt in seiner Anzeigeliste enthalten – das Anzeigeobjekt, das die geladene SWF- oder Grafikdatei darstellt. Wenn Sie der Anzeigeliste ein Loader-Objekt hinzufügen (wie im folgenden Code), haben Sie der Anzeigeliste auch das geladene untergeordnete Anzeigeobjekt hinzugefügt (nachdem es geladen wurde):

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

Wenn die SWF-Datei oder das Bild geladen ist, können Sie das geladene Anzeigeobjekt in einen anderen Anzeigeobjektcontainer verschieben, z. B. in das DisplayObjectContainer-Objekt `container` im folgenden Beispiel:

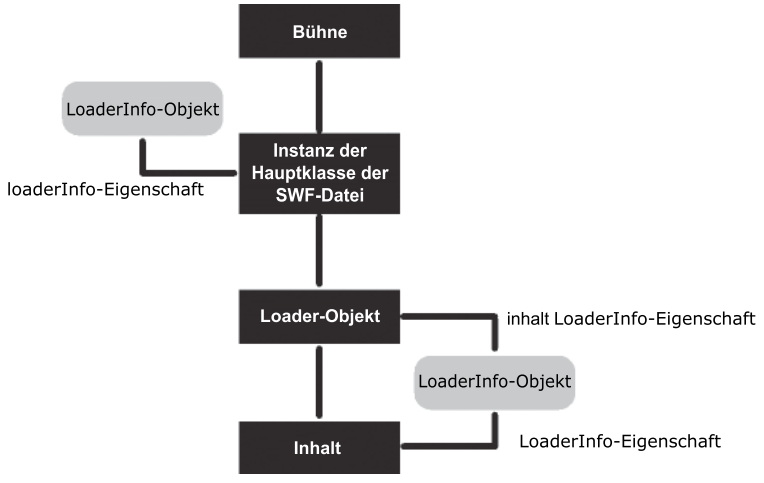
```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
    container.addChild(pictLdr.content);
}
```

Überwachen des Ladefortschritts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn das Laden der Datei gestartet wurde, wird ein LoaderInfo-Objekt erstellt. Ein LoaderInfo-Objekt enthält Informationen wie den Ladefortschritt, die URLs des ladenden und des geladenen Objekts, die Gesamtanzahl der Bytes für das Medium und die nominelle Höhe und Breite des Mediums. Ein LoaderInfo-Objekt löst darüber hinaus Ereignisse zur Überwachung des Ladefortschritts aus.

Das folgende Diagramm zeigt die verschiedenen Einsatzmöglichkeiten des LoaderInfo-Objekts – für die Instanz der Hauptklasse der SWF-Datei, für ein Loader-Objekt und für ein vom Loader-Objekt geladenes Objekt:



Auf das LoaderInfo-Objekt kann als Eigenschaft des Loader-Objekts und des geladenen Anzeigeobjekts zugegriffen werden. Sobald das Laden begonnen hat, kann über die Eigenschaft `contentLoaderInfo` des Loader-Objekts auf das LoaderInfo-Objekt zugegriffen werden. Wenn das Anzeigeobjekt vollständig geladen ist, kann über die Eigenschaft `loaderInfo` des Anzeigeobjekts als Eigenschaft des geladenen Anzeigeobjekts auf das LoaderInfo-Objekt zugegriffen werden. Die Eigenschaft `loaderInfo` des geladenen Anzeigeobjekts verweist auf das gleiche LoaderInfo-Objekt wie die Eigenschaft `contentLoaderInfo` des Loader-Objekts. Anders ausgedrückt, das geladene Objekt und das Loader-Objekt, das es geladen hat, verwenden dasselbe LoaderInfo-Objekt.

Um auf die Eigenschaften des geladenen Inhalts zugreifen zu können, müssen Sie dem LoaderInfo-Objekt einen Ereignis-Listener hinzufügen, wie im folgenden Code dargestellt:

```
import flash.display.Loader;
import flash.display.Sprite;
import flash.events.Event;

var ldr:Loader = new Loader();
var urlReq:URLRequest = new URLRequest("Circle.swf");
ldr.load(urlReq);
ldr.contentLoaderInfo.addEventListener(Event.COMPLETE, loaded);
addChild(ldr);

function loaded(event:Event):void
{
    var content:Sprite = event.target.content;
    content.scaleX = 2;
}
```

Weitere Informationen finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Festlegen des Ladekontexts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie eine externe Datei über die `load()`- oder `loadBytes()`-Methode der Loader-Klasse in Flash Player oder AIR laden, können Sie optional einen `context`-Parameter angeben. Bei diesem Parameter handelt es sich um ein LoaderContext-Objekt.

Die LoaderContext-Klasse umfasst drei Eigenschaften, mit denen Sie den Kontext definieren können, wie der geladene Inhalt verwendet werden kann:

- `checkPolicyFile`: Verwenden Sie diese Eigenschaft nur beim Laden einer Bilddatei (nicht beim Laden einer SWF-Datei). Wenn Sie diese Eigenschaft auf `true` festlegen, sucht der Loader auf dem Ursprungsserver nach einer Richtliniendatei (siehe „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125). Dies ist nur dann notwendig, wenn Inhalte aus anderen Domänen stammen als die SWF-Datei, in der das Loader-Objekt enthalten ist. Wenn der Server Zugriff auf die Loader-Domäne gewährt, kann ActionScript aus SWF-Dateien in der Loader-Domäne auf Daten im geladenen Bild zugreifen. Anders ausgedrückt, Sie können mit dem Befehl `BitmapData.draw()` auf Daten im geladenen Bild zugreifen.

  Beachten Sie, dass eine SWF-Datei aus einer anderen Domäne als das Loader-Objekt `Security.allowDomain()` aufrufen kann, um eine bestimmte Domäne zuzulassen.

- `securityDomain`: Verwenden Sie diese Eigenschaft nur beim Laden einer SWF-Datei (nicht beim Laden eines Bilds). Dies ist nur dann notwendig, wenn eine SWF-Datei aus einer anderen Domäne stammt als die Datei, die das Loader-Objekt enthält. Wenn Sie diese Option angeben, sucht Flash Player nach einer Richtliniendatei. Ist eine vorhanden, können SWF-Dateien aus Domänen, die in der domänenübergreifenden Richtliniendatei enthalten sind, auf den geladenen SWF-Inhalt verweisen (Cross-Scripting). Sie können `flash.system.SecurityDomain.currentDomain` als Parameter angeben.

- `applicationDomain`: Verwenden Sie diese Eigenschaft nur beim Laden einer SWF-Datei, die in ActionScript 3.0 geschrieben wurde (nicht beim Laden eines Bilds oder einer SWF-Datei, die in ActionScript 1.0 oder 2.0 geschrieben wurde). Beim Laden der Datei können Sie festlegen, ob die Datei in die gleiche Anwendungsdomäne wie die des Loader-Objekts aufgenommen werden soll, indem Sie den Parameter `applicationDomain` auf `flash.system.ApplicationDomain.currentDomain` einstellen. Durch Einfügen der geladenen SWF-Datei in die gleiche Anwendungsdomäne können Sie direkt auf die zugehörigen Klassen zugreifen. Dies ist insbesondere beim Laden einer SWF-Datei von Nutzen, die eingebettete Medien enthält, auf die Sie über denen zugewiesenen Klassennamen zugreifen können. Weitere Informationen finden Sie unter „Verwenden von Anwendungsdomänen“ auf Seite 157.

Im Folgenden ist ein Beispiel für die Suche nach einer Richtliniendatei aufgeführt, wenn eine Bitmap aus einer anderen Domäne geladen wird:

```
var context:LoaderContext = new LoaderContext();
context.checkPolicyFile = true;
var urlReq:URLRequest = new URLRequest("http://www.[your_domain_here].com/photo11.jpg");
var ldr:Loader = new Loader();
ldr.load(urlReq, context);
```

Im Folgenden ist ein Beispiel für die Suche nach einer Richtliniendatei aufgeführt, wenn eine SWF-Datei aus einer anderen Domäne geladen wird, um diese Datei in der gleichen Sicherheits-Sandbox wie das Loader-Objekt zu platzieren. Darüber hinaus fügt der Code die Klassen in der geladenen SWF-Datei der gleichen Anwendungsdomäne wie das Loader-Objekt hinzu:

```
var context:LoaderContext = new LoaderContext();
context.securityDomain = SecurityDomain.currentDomain;
context.applicationDomain = ApplicationDomain.currentDomain;
var urlReq:URLRequest = new URLRequest("http://www.[your_domain_here].com/library.swf");
var ldr:Loader = new Loader();
ldr.load(urlReq, context);
```

Weitere Informationen finden Sie in der Beschreibung der LoaderContext-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Laden von SWF-Dateien in AIR für iOS

Adobe AIR 3.6 und höher – nur iOS

Auf iOS-Geräten ist das Laden und Kompilieren von Code zur Laufzeit beschränkt. Aufgrund dieser Beschränkungen weist das Laden externer SWF-Dateien in Ihre Anwendung einige notwendige Unterschiede auf:

- Alle SWF-Dateien mit ActionScript-Code müssen im Anwendungspaket enthalten sein. SWF-Dateien, die Code enthalten, können nicht aus einer externen Quelle (z. B. über ein Netzwerk) geladen werden. Als Teil des Verpackens der Anwendung wird der gesamte ActionScript-Code in allen SWF-Dateien des Anwendungspakets in nativen Code für iOS-Geräte kompiliert.
- Sie können eine SWF-Datei nicht laden, entladen und dann erneut laden. Wenn Sie dies versuchen, tritt ein Fehler auf.
- Beim Laden in den Arbeitsspeicher und bei anschließendem Entladen ist das Verhalten identisch mit dem in Desktop-Plattformen. Wenn Sie eine SWF-Datei laden und dann entladen, werden alle visuellen Elemente in der SWF-Datei aus dem Arbeitsspeicher entfernt. Alle Klassenverweise zu einer ActionScript-Klasse in der geladenen SWF-Datei bleiben jedoch im Arbeitsspeicher erhalten und können über den ActionScript-Code aufgerufen werden.
- Alle geladenen SWF-Dateien müssen in dieselbe Anwendungsdomäne wie die Haupt-SWF-Datei geladen werden. Dies ist nicht das Standardverhalten. Daher müssen Sie für jede zu ladende SWF-Datei ein LoaderContext-Objekt erstellen, das die Hauptanwendungsdomäne angibt, und das LoaderContext-Objekt an den Aufruf der Loader.load()-Methode übergeben. Wenn Sie versuchen, die SWF-Datei in eine andere Anwendungsdomäne als die Anwendungsdomäne der Haupt-SWF-Datei zu laden, tritt ein Fehler auf. Dies trifft auch zu, wenn die geladene SWF-Datei nur visuelle Elemente und keinen ActionScript-Code enthält.

  Das folgende Beispiel zeigt den Code, mit dem eine SWF-Datei aus dem Anwendungspaket in die Anwendungsdomäne der Haupt-SWF-Datei geladen wird:

  ```
  var loader:Loader = new Loader();
  var url:URLRequest = new URLRequest("swfs/SecondarySwf.swf");
  var loaderContext:LoaderContext = new LoaderContext(false, ApplicationDomain.currentDomain,
  null);
  loader.load(url, loaderContext);
  ```

Eine SWF-Datei, die nur Elemente und keinen Code enthält, kann aus dem Anwendungspaket oder über ein Netzwerk geladen werden. In jedem Fall muss die SWF-Datei weiterhin in die Hauptanwendungsdomäne geladen sein.

Bei AIR-Versionen vor AIR 3.6 wird der gesamte Code bei der Kompilierung aus den SWF-Dateien (außer der Haupt-SWF-Datei der Anwendung) entfernt. SWF-Dateien, die nur visuelle Elemente (jedoch keinen Code) enthalten, können in das Anwendungspaket eingebunden und zur Laufzeit geladen werden. Wenn Sie eine SWF-Datei mit ActionScript-Code laden, tritt ein Fehler auf. Daraufhin wird in der Anwendung eine Fehlermeldung zu nicht kompiliertem ActionScript-Code angezeigt.

Siehe auch

Packaging and loading multiple SWFs in AIR apps on iOS

Verwenden der ProLoader- und ProLoaderInfo-Klassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher; erfordert Flash Professional CS5.5

Als Hilfe zum Vorausladen von gemeinsam genutzten Remote-Bibliotheken (RSL, Remote Shared Library) wurden in Flash Professional CS5.5 die fl.display.ProLoader- und fl.display.ProLoaderInfo-Klassen eingeführt. Diese Klassen ähneln den flash.display.Loader- und flash.display.LoaderInfo-Klassen, bieten jedoch ein einheitlicheres Verhalten beim Laden.

Insbesondere ProLoader ist zum Laden von SWF-Dateien nützlich, die das Text Layout Framework (TLF) mit dem RSL-Vorausladen verwenden. SWF-Dateien, die andere SWF-Dateien oder SWZ-Dateien vorausladen, wie beispielsweise TLF, erfordern zur Laufzeit eine rein interne SWF-Wrapper-Datei. Da die SWF-Wrapper-Datei zusätzliche Komplexität verursacht, kann ein unerwartetes Verhalten auftreten. ProLoader löst diese Probleme bezüglich der Komplexität und lädt die Dateien wie normale SWF-Dateien. Die von der ProLoader-Klasse bereitgestellte Lösung ist für den Benutzer transparent und erfordert keine besondere Verarbeitung in ActionScript. Außerdem wird regulärer SWF-Inhalt mit ProLoader korrekt geladen.

In Flash Professional ab CS5.5 können Sie alle Vorkommen der Loader-Klasse problemlos durch die ProLoader-Klasse ersetzen. Exportieren Sie Ihre Anwendung dann in Flash Player 10.2 oder höher, damit ProLoader auf die erforderliche ActionScript-Funktionalität zugreifen kann. Sie können ProLoader auch für frühere Versionen von Flash Player verwenden, die ActionScript 3.0 unterstützen. Das volle Funktionsspektrum von ProLoader steht jedoch erst ab Flash Player 10.2 zur Verfügung. Verwenden Sie mit TLF in Flash Professional CS5.5 oder höher immer ProLoader. ProLoader wird in anderen Umgebungen außer Flash Professional nicht benötigt.

Wichtig: *Für SWF-Dateien, die in Flash Professional CS5.5 oder höher veröffentlicht werden, können Sie immer die fl.display.ProLoader- und fl.display.ProLoaderInfo-Klassen anstelle von flash.display.Loader und flash.display.LoaderInfo verwenden.*

Von der ProLoader-Klasse behobene Probleme

Die ProLoader-Klasse behebt Probleme, die mit der älteren Loader-Klasse nicht bewältigt werden konnten. Diese Probleme sind auf das RSL-Vorausladen von TLF-Bibliotheken zurückzuführen. Insbesondere betrifft dies SWF-Dateien, die ein Loader-Objekt verwenden, um andere SWF-Dateien zu laden. Zu den behobenen Problemen gehören:

- **Die Skripterstellung zwischen der ladenden Datei und der geladenen Datei verhält sich nicht wie erwartet.** Die ProLoader-Klasse legt die ladende SWF-Datei automatisch als übergeordnetes Objekt der geladenen SWF-Datei fest. Deshalb erfolgt die Kommunikation von der ladenden SWF-Datei direkt an die geladene SWF-Datei.
- **Die SWF-Anwendung muss den Ladevorgang aktiv verwalten.** Dazu müssen zusätzliche Ereignisse implementiert werden, wie `added`, `removed`, `addedToStage` und `removedFromStage`. Wenn Ihre Anwendung für Flash Player 10.2 oder höher vorgesehen ist, kann diese zusätzliche Arbeit mithilfe von ProLoader vermieden werden.

Aktualisieren von Code zur Verwendung von ProLoader anstelle von Loader

Da ProLoader der Loader-Klasse ähnelt, können Sie die beiden Klassen in Ihrem Code ganz einfach austauschen. Das folgende Beispiel zeigt, wie vorhandener Code zur Verwendung der neuen Klasse aktualisiert wird:

```
import flash.display.Loader;
import flash.events.Event;
var l:Loader = new Loader();

addChild(l);
l.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete);
l.load("my.swf");
function loadComplete(e:Event) {
    trace('load complete!');
}
```

Dieser Code kann folgendermaßen für die Verwendung von ProLoader geändert werden:

```
import fl.display.ProLoader;
import flash.events.Event;
var l:ProLoader = new ProLoader();

addChild(l);
l.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete);
l.load("my.swf");
function loadComplete(e:Event) {
    trace('load complete!');
}
```

Beispiel für ein Anzeigeobjekt: SpriteArranger

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Beispielanwendung „SpriteArranger“ baut auf der Beispielanwendung „Geometric Shapes“ auf, die im *ActionScript 3.0 – Arbeitshandbuch* beschrieben wird.

Die Beispielanwendung „SpriteArranger“ verdeutlicht verschiedene Konzepte beim Verwenden von Anzeigeobjekten:

- Erweitern von Anzeigeobjektklassen
- Hinzufügen von Objekten zur Anzeigeliste
- Anordnen von Anzeigeobjekten auf Ebenen und Arbeiten mit Anzeigeobjektcontainern
- Reagieren auf Anzeigeobjektereignisse
- Arbeiten mit den Eigenschaften und Methoden von Anzeigeobjekten

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der SpriteArranger-Anwendung befinden sich im Ordner „Examples/SpriteArranger“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
SpriteArranger.mxml oder SpriteArranger.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/SpriteArranger/CircleSprite.as	Eine Klasse, die ein Sprite-Objekt definiert, das einen Kreis auf dem Bildschirm anzeigt.
com/example/programmingas3/SpriteArranger/DrawingCanvas.as	Eine Klasse, die eine Leinwand definiert, die der Anzeigeobjektcontainer ist, in dem die GeometricSprite-Objekte enthalten ist.
com/example/programmingas3/SpriteArranger/SquareSprite.as	Eine Klasse, die ein Sprite-Objekt definiert, das ein Quadrat auf dem Bildschirm anzeigt.
com/example/programmingas3/SpriteArranger/TriangleSprite.as	Eine Klasse, die ein Sprite-Objekt definiert, das ein Dreieck auf dem Bildschirm anzeigt.
com/example/programmingas3/SpriteArranger/GeometricSprite.as	Eine Klasse, die das Sprite-Objekt erweitert und zum Definieren einer Form auf dem Bildschirm verwendet wird. Diese Klasse wird von CircleSprite, SquareSprite und TriangleSprite erweitert.
com/example/programmingas3/geometricshapes/IGeometricShape.as	Die Methoden, mit denen die grundlegenden Schnittstelle definiert wird, die von allen GeometricShapes-Klassen implementiert werden müssen.
com/example/programmingas3/geometricshapes/IPolygon.as	Die Methoden, mit denen eine Schnittstelle definiert wird, die von allen GeometricShapes-Klassen mit mehreren Seiten implementiert werden müssen.
com/example/programmingas3/geometricshapes/RegularPolygon.as	Eine geometrische Form, deren gleich lange Seiten symmetrisch um den Mittelpunkt der Form positioniert sind.
com/example/programmingas3/geometricshapes/Circle.as	Eine geometrische Form, die einen Kreis definiert.
com/example/programmingas3/geometricshapes/EquilateralTriangle.as	Eine Unterklasse von RegularPolygon, die ein Dreieck definiert, dessen Seiten die gleiche Länge aufweisen.
com/example/programmingas3/geometricshapes/Square.as	Eine Unterklasse von RegularPolygon, die ein Rechteck definiert, dessen vier Seiten die gleiche Länge aufweisen.
com/example/programmingas3/geometricshapes/GeometricShapeFactory.as	Eine Klasse, die eine „Factory-Methode“ zum Erstellen von Formen eines bestimmten Typs und einer bestimmten Größe enthält.

Definieren der SpriteArranger-Klassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Anwendung „SpriteArranger“ können Benutzer der „Leinwand“ auf dem Bildschirm verschiedene Anzeigeobjekte hinzufügen.

Die DrawingCanvas-Klasse definiert einen Zeichenbereich (einen Anzeigeobjektcontainer), dem der Benutzer auf dem Bildschirm anzuzeigende Formen hinzufügen kann. Diese Bildschirm-Formen sind Instanzen einer der Unterklassen der GeometricSprite-Klasse.

DrawingCanvas-Klasse

In Flex müssen alle untergeordneten Anzeigeobjekte, die einem Container-Objekt hinzugefügt werden, einer Klasse angehören, die ein Nachfolger der mx.core.UIComponent-Klasse ist. Diese Anwendung fügt eine Instanz der DrawingCanvas-Klasse als untergeordnetes Element eines mx.containers.VBox-Objekts hinzu, wie im MXML-Code in der Datei „SpriteArranger.mxml" definiert. Diese Vererbung ist in der Deklaration DrawingCanvas-Klasse wie folgt definiert:

```
public class DrawingCanvas extends UIComponent
```

Die UIComponent-Klasse erbt von der DisplayObject-, der DisplayObjectContainer- und der Sprite-Klasse und der Code in der DrawingCanvas-Klasse verwendet Methoden und Eigenschaften dieser Klassen.

Die DrawingCanvas-Klasse erweitert die Sprite-Klasse. Diese Vererbung ist in der Deklaration der DrawingCanvas-Klasse wie folgt definiert:

```
public class DrawingCanvas extends Sprite
```

Die Sprite-Klasse ist eine Unterklasse der DisplayObjectContainer- und der DisplayObject-Klasse. In der DrawingCanvas-Klasse werden Methoden und Eigenschaften dieser Klassen verwendet.

Die `DrawingCanvas()`-Konstruktormethode richtet ein Rectangle-Objekt namens `bounds` ein, bei dem es sich um eine Eigenschaft handelt, die später zum Zeichnen der Leinwandkontur verwendet wird. Dann ruft sie die `initCanvas()`-Methode auf:

```
this.bounds = new Rectangle(0, 0, w, h);
initCanvas(fillColor, lineColor);
```

Wie das folgende Beispiel zeigt, definiert die Methode `initCanvas()` verschiedene Eigenschaften des DrawingCanvas-Objekts, die als Argumente an die Konstruktorfunktion übergeben werden:

```
this.lineColor = lineColor;
this.fillColor = fillColor;
this.width = 500;
this.height = 200;
```

Die Methode `initCanvas()` ruft dann die Methode `drawBounds()` auf, die mit der Eigenschaft `graphics` der DrawingCanvas-Klasse die Leinwand zeichnet. Die Eigenschaft `graphics` wird von der Shape-Klasse geerbt.

```
this.graphics.clear();
this.graphics.lineStyle(1.0, this.lineColor, 1.0);
this.graphics.beginFill(this.fillColor, 1.0);
this.graphics.drawRect(bounds.left - 1,
                       bounds.top - 1,
                       bounds.width + 2,
                       bounds.height + 2);
this.graphics.endFill();
```

Die folgenden zusätzlichen Methoden der DrawingCanvas-Klasse werden basierend auf Interaktionen des Benutzers mit der Anwendung aufgerufen:

- Methoden `addShape()` und `describeChildren()`; siehe „Hinzufügen von Anzeigeobjekten zur Leinwand" auf Seite 220
- Methoden `moveToBack()`, `moveDown()`, `moveToFront()` und `moveUp()`, siehe „Neuanordnen der Anzeigeobjektebenen" auf Seite 223
- Methode `onMouseUp()`; siehe „Klicken und Ziehen von Anzeigeobjekten" auf Seite 222

GeometricSprite-Klasse und ihre Unterklassen

Jedes Anzeigeobjekt, das ein Benutzer der Leinwand hinzufügen kann, ist eine Instanz einer der folgenden Unterklassen der GeometricSprite-Klasse:

- CircleSprite
- SquareSprite
- TriangleSprite

Die GeometricSprite-Klasse erweitert die flash.display.Sprite-Klasse:

```
public class GeometricSprite extends Sprite
```

Die GeometricSprite-Klasse definiert verschiedene Eigenschaften, die allen GeometricSprite-Objekten gemein sind. Diese werden in der Konstruktorfunktion basierend auf Parametern eingestellt, die an die Funktion übergeben werden. Zum Beispiel:

```
this.size = size;
this.lineColor = lColor;
this.fillColor = fColor;
```

Die Eigenschaft `geometricShape` der GeometricSprite-Klasse definiert eine IGeometricShape-Schnittstelle, die wiederum die mathematischen Eigenschaften der Form, aber nicht ihre visuellen Eigenschaften festgelegt. Die Klassen, die die IGeometricShape-Schnittstelle implementieren, werden in der GeometricShapes-Beispielanwendung definiert, die im *ActionScript 3.0 – Arbeitshandbuch* beschrieben wird.

Die GeometricSprite-Klasse definiert die Methode `drawShape()`, die darüber hinaus in den Überschreibungsdefinitionen jeder Unterklasse von GeometricSprite weiter definiert wird. Weitere Informationen finden Sie im nachfolgenden Abschnitt „Hinzufügen von Anzeigeobjekten zur Leinwand“.

Darüber hinaus stellt die GeometricSprite-Klasse folgende Methoden zur Verfügung:

- Methoden `onMouseDown()` und `onMouseUp()`; siehe „Klicken und Ziehen von Anzeigeobjekten“ auf Seite 222
- Methoden `showSelected()` und `hideSelected()`; siehe „Klicken und Ziehen von Anzeigeobjekten“ auf Seite 222

Hinzufügen von Anzeigeobjekten zur Leinwand

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Klickt ein Benutzer auf die Schaltfläche „Form hinzufügen“, ruft in die Anwendung die Methode `addShape()` der DrawingCanvas-Klasse auf. Sie instanziiert ein neues GeometricSprite, indem sie die geeignete Konstruktorfunktion einer der GeometricSprite-Unterklassen aufruft. Dies wird im folgenden Beispiel gezeigt:

```
public function addShape(shapeName:String, len:Number):void
{
    var newShape:GeometricSprite;
    switch (shapeName)
    {
        case "Triangle":
            newShape = new TriangleSprite(len);
            break;

        case "Square":
            newShape = new SquareSprite(len);
            break;

        case "Circle":
            newShape = new CircleSprite(len);
            break;
    }
    newShape.alpha = 0.8;
    this.addChild(newShape);
}
```

Jede Konstruktormethode ruft die Methode `drawShape()` auf, die wiederum die `graphics`-Eigenschaft der Klasse verwendet (von der Sprite-Klasse geerbt), um die entsprechende Vektorgrafik zu zeichnen. Beispielsweise enthält die `drawShape()`-Methode der CircleSprite-Klasse den folgenden Code:

```
this.graphics.clear();
this.graphics.lineStyle(1.0, this.lineColor, 1.0);
this.graphics.beginFill(this.fillColor, 1.0);
var radius:Number = this.size / 2;
this.graphics.drawCircle(radius, radius, radius);
```

Die vorletzte Zeile der `addShape()`-Funktion stellt die `alpha`-Eigenschaft des Anzeigeobjekts ein (von der DisplayObject-Klasse geerbt). Jedes der Leinwand hinzugefügte Anzeigeobjekt erscheint etwas transparent und lässt den Benutzer sehen, was sich dahinter befindet.

Die letzte Zeile der `addChild()`-Methode fügt das neue Anzeigeobjekt zur Child-Liste der Instanz der DrawingCanvas-Klasse hinzu, die sich bereits in der Anzeigeliste befindet. Dadurch wird das neue Anzeigeobjekt auf der Bühne angezeigt.

Die Schnittstelle dieser Anwendung umfasst zwei Textfelder, `selectedSpriteTxt` und `outputTxt`. Die Texteigenschaften dieser Textfelder werden mit Informationen zu den GeometricSprite-Objekten aktualisiert, die der Leinwand hinzugefügt oder vom Benutzer ausgewählt wurden. Die GeometricSprite-Klasse verarbeitet diese Aufgabe zum Melden von Informationen durch Überschreiben der `toString()`-Methode. Dies wird im folgenden Beispiel gezeigt:

```
public override function toString():String
{
    return this.shapeType + " of size " + this.size + " at " + this.x + ", " + this.y;
}
```

Die Eigenschaft `shapeType` wird in der Konstruktormethode jeder GeometricSprite-Unterklasse auf einen geeigneten Wert gesetzt. Beispielsweise könnte die Methode `toString()` den folgenden Wert für eine CircleSprite-Instanz zurückgeben, die der DrawingCanvas-Instanz vor kurzem hinzugefügt wurde:

```
Circle of size 50 at 0, 0
```

Die `describeChildren()`-Methode der DrawingCanvas-Klasse durchläuft die Child-Liste der Leinwand und verwendet die Eigenschaft `numChildren` (von der DisplayObjectContainer-Klasse geerbt), um den Grenzwert für die `for`-Schleife festzulegen. Dies erstellt eine Zeichenfolge, die alle untergeordneten Elemente aufführt:

```
var desc:String = "";
var child:DisplayObject;
for (var i:int=0; i < this.numChildren; i++)
{
child = this.getChildAt(i);
desc += i + ": " + child + '\n';
}
```

Die resultierende Zeichenfolge dient zum Einstellen der Eigenschaft `text` des Textfeldes `outputTxt`.

Klicken und Ziehen von Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Klickt der Benutzer auf eine GeometricSprite-Instanz, ruft die Anwendung die `onMouseDown()`-Ereignisprozedur auf. Wie der folgende Code zeigt, ist diese Ereignisprozedur in der Konstruktorfunktion der GeometricSprite-Klasse zur Überwachung auf eine gedrückte Maustaste eingestellt:

```
this.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
```

Die `onMouseDown()`-Methode ruft daraufhin die `showSelected()`-Methode des GeometricSprite-Objekts auf. Wird diese Methode das erste Mal für ein Objekt aufgerufen, erstellt sie ein neues Shape-Objekt namens `selectionIndicator` und verwendet die Eigenschaft `graphics` des Shape-Objekts zum Zeichnen eines roten Markierungsrechtecks. Dies wird im folgenden Beispiel gezeigt:

```
this.selectionIndicator = new Shape();
this.selectionIndicator.graphics.lineStyle(1.0, 0xFF0000, 1.0);
this.selectionIndicator.graphics.drawRect(-1, -1, this.size + 1, this.size + 1);
this.addChild(this.selectionIndicator);
```

Wird die Methode `onMouseDown()` nicht das erste Mal aufgerufen, stellt sie einfach die Eigenschaft `visible` der Form `selectionIndicator` (von der DisplayObject-Klasse geerbt) wie folgt ein:

```
this.selectionIndicator.visible = true;
```

Die `hideSelected()`-Methode blendet die `selectionIndicator`-Form des zuvor ausgewählten Objekts aus, indem sie deren Eigenschaft `visible` auf `false` einstellt.

Außerdem ruft die Ereignisprozedurmethode `onMouseDown()` die `startDrag()`-Methode auf (von der Sprite-Klasse geerbt), die den folgenden Code enthält:

```
var boundsRect:Rectangle = this.parent.getRect(this.parent);
boundsRect.width -= this.size;
boundsRect.height -= this.size;
this.startDrag(false, boundsRect);
```

Dadurch kann der Benutzer das ausgewählte Objekt auf der Leinwand verschieben, und zwar innerhalb der Grenzen, die durch das `boundsRect`-Rechteck festgelegt werden.

Lässt der Benutzer die Maustaste los, wird das `mouseUp`-Ereignis ausgelöst. Die Konstruktormethode von DrawingCanvas richtet den folgenden Ereignis-Listener ein:

```
this.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
```

Dieser Ereignis-Listener ist für das DrawingCanvas-Objekt und nicht für einzelne GeometricSprite-Objekte eingestellt. Der Grund hierfür ist, dass sich ein gezogenes GeometricSprite-Objekt beim Loslassen der Maustaste hinter einem anderen Anzeigeobjekt (einem anderen GeometricSprite-Objekt) befinden könnte. Dann würde das Anzeigeobjekt im Vordergrund das MouseUp-Ereignis erhalten, das vom Benutzer gezogene Anzeigeobjekt jedoch nicht. Das Hinzufügen dessen Listeners zum DrawingCanvas-Objekt stellt sicher, dass das Ereignis immer verarbeitet wird.

Die `onMouseUp()`-Methode ruft die `onMouseUp()`-Methode des GeometricSprite-Objekts auf, die wiederum die `stopDrag()`-Methode des GeometricSprite-Objekts aufruft.

Neuanordnen der Anzeigeobjektebenen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Benutzeroberfläche der Anwendung umfasst Schaltflächen mit den Beschriftungen „Move Back“, „Move Down“, „Move Up“ und „Move to Front“. Wenn der Benutzer auf eine dieser Schaltflächen klickt, ruft die Anwendung die entsprechende Methode der DrawingCanvas-Klasse auf: `moveToBack()`, `moveDown()`, `moveUp()` oder `moveToFront()`. Beispielsweise enthält die `moveToBack()`-Methode den folgenden Code:

```
public function moveToBack(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, 0);
    }
}
```

Diese Methode verwendet die `setChildIndex()`-Methode (von der DisplayObjectContainer-Klasse geerbt), um das Anzeigeobjekt an Indexposition 0 in der Child-Liste der DrawingCanvas-Instanz (`this`) zu positionieren.

Die `moveDown()`-Methode arbeitet ähnlich, außer dass sie die Indexposition des Anzeigeobjekts in der Child-Liste der DrawingCanvas-Instanz um 1 verringert:

```
public function moveDown(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, index - 1);
    }
}
```

Die Methoden `moveUp()` und `moveToFront()` arbeiten ähnlich wie die Methoden `moveToBack()` und `moveDown()`.

Kapitel 11: Verwenden von geometrischen Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das flash.geom-Paket enthält Klassen, mit denen geometrische Objekte definiert werden, z. B. Punkte, Rechtecke und Transformationsmatrizen. Mit diesen Klassen können Sie die Eigenschaften von Objekten festlegen, die in anderen Klassen verwendet werden.

Verwandte Hilfethemen

flash.geom-Paket

Grundlagen von geometrischen Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das flash.geom-Paket enthält Klassen, mit denen geometrische Objekte definiert werden, z. B. Punkte, Rechtecke und Transformationsmatrizen. Diese Klassen müssen nicht notwendigerweise selbst Funktionen bereitstellen. Mithilfe dieser Klassen werden jedoch die Eigenschaften von Objekten definiert, die in anderen Klassen verwendet werden.

Alle Geometrieklassen beruhen auf der Vorstellung, dass Positionen auf dem Bildschirm in einer zweidimensionalen Ebene angegeben werden. Der Bildschirm wird als zweidimensionales Diagramm mit einer horizontalen x-Achse und einer vertikalen y-Achse behandelt. Jede Position (bzw. jeder *Punkt*) auf dem Bildschirm kann als Paar aus einem x-Wert und einem y-Wert, den *Koordinaten* dieser Position, dargestellt werden.

Jedes Anzeigeobjekt, auch die Bühne, hat einen eigenen *Koordinatenraum*. Der Koordinatenraum bildet die Grundlage, auf der das Objekt die Position von untergeordneten Anzeigeobjekten, Zeichnungen und so weiter bestimmt. Der *Ursprung* befindet sich an den Koordinaten 0, 0 (dem Schnittpunkt der x- und y-Achsen). Er wird an der oberen linken Ecke des Anzeigeobjekts platziert. Diese Ursprungsposition gilt stets für die Bühne, aber nicht unbedingt für andere Anzeigeobjekte. Werte auf der x-Achse nehmen nach rechts zu und nach links ab. Bei Positionen links vom Ursprung ist die x-Koordinate negativ. Im Gegensatz zu herkömmlichen Koordinatensystemen nehmen die Koordinatenwerte auf der y-Achse in der Flash-Laufzeit nach unten auf dem Bildschirm zu und nach oben ab. Werte oberhalb des Ursprungs haben einen negativen y-Koordinatenwert. Da die obere linke Ecke der Bühne den Ursprung ihres Koordinatenraums bildet, haben die meisten Objekte auf der Bühne eine x-Koordinate größer als 0 und kleiner als die Breite der Bühne. Gleichermaßen haben die Objekte eine y-Koordinate größer als 0 und kleiner als die Höhe der Bühne.

Mithilfe von Instanzen der Point-Klasse können Sie einzelne Punkte in einem Koordinatenraum angeben. Sie können eine Rectangle-Instanz erstellen, um einen rechteckigen Bereich in einem Koordinatenraum anzugeben. Erfahrene Benutzer können mithilfe einer Matrix-Instanz mehrere oder komplexe Transformationen auf ein Anzeigeobjekt anwenden. Viele einfache Transformationen, z. B. Drehung, Positionierung und Skalierung, können über die Eigenschaften eines Anzeigeobjekts direkt auf das Objekt angewendet werden. Weitere Informationen zu Transformationen mithilfe der Eigenschaften von Anzeigeobjekten finden Sie unter „Bearbeiten von Anzeigeobjekten“ auf Seite 185.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Geometriebegriffe aufgeführt:

Kartesische Koordinaten Koordinaten werden in der Regel als Zahlenpaar geschrieben (z. B. 5, 12 oder 17, -23). Mit den beiden Zahlen wird jeweils die x-Koordinate und die y-Koordinate angegeben.

Koordinatenraum Das Koordinatendiagramm eines Anzeigeobjekts, in dem die jeweils untergeordneten Elemente positioniert sind.

Ursprung Der Punkt in einem Koordinatenraum, an dem die x- und y-Achsen sich schneiden. Dieser Punkt hat die Koordinaten (0, 0).

Punkt Eine einzelne Position in einem Koordinatenraum. In dem in ActionScript verwendeten 2D-Koordinatensystem wird der Punkt von seiner Position entlang der x-Achse und der y-Achse (den Punktkoordinaten) definiert.

Registrierungspunkt Der Ursprung (0, 0) des Koordinatenraums eines Anzeigeobjekts.

Skalieren Die Größe eines Objekts im Verhältnis zu seiner Originalgröße. Beim Skalieren eines Objekts wird die Größe des Objekts geändert, indem das Objekt vergrößert oder verkleinert wird.

Versetzen Umwandeln der Koordinaten eines Punkts von einem Koordinatenraum in einen anderen Koordinatenraum.

Transformation Ändern der visuellen Merkmale einer Grafik, beispielsweise durch Drehen, Ändern der Größe, Neigen, Verzerren oder Ändern der Farbe eines Objekts.

X-Achse Die horizontale Achse des in ActionScript verwendeten zweidimensionalen Koordinatensystems.

Y-Achse Die vertikale Achse des in ActionScript verwendeten zweidimensionalen Koordinatensystems.

Verwenden von Point-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Point-Objekt definiert ein Paar kartesischer Koordinaten. Es gibt eine Position in einem zweidimensionalen Koordinatensystem an, in dem *x* die horizontale Achse und *y* die vertikale Achse darstellt.

Ein Point-Objekt wird durch Festlegen der entsprechenden `x`-Eigenschaft und `y`-Eigenschaft definiert:

```
import flash.geom.*;
var pt1:Point = new Point(10, 20); // x == 10; y == 20
var pt2:Point = new Point();
pt2.x = 10;
pt2.y = 20;
```

Ermitteln des Abstands zwischen zwei Punkten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der `distance()`-Methode der Point-Klasse können Sie den Abstand zwischen zwei Punkten in einem Koordinatenraum ermitteln. Mit dem folgenden Code wird beispielsweise der Abstand zwischen den Registrierungspunkten der beiden Anzeigeobjekte `circle1` und `circle2` im gleichen Anzeigeobjektcontainer ermittelt:

```
import flash.geom.*;
var pt1:Point = new Point(circle1.x, circle1.y);
var pt2:Point = new Point(circle2.x, circle2.y);
var distance:Number = Point.distance(pt1, pt2);
```

Versetzen von Koordinatenräumen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn zwei Anzeigeobjekte in verschiedenen Anzeigeobjektcontainern enthalten sind, können sie sich in verschiedenen Koordinatenräumen befinden. Mithilfe der `localToGlobal()`-Methode der DisplayObject-Klasse können Sie die Koordinaten in den gleichen (globalen) Koordinatenraum versetzen, d. h. in den Koordinatenraum der Bühne. Mit dem folgenden Code wird beispielsweise der Abstand zwischen den Registrierungspunkten der beiden Anzeigeobjekte `circle1` und `circle2` in zwei verschiedenen Anzeigeobjektcontainern ermittelt:

```
import flash.geom.*;
var pt1:Point = new Point(circle1.x, circle1.y);
pt1 = circle1.localToGlobal(pt1);
var pt2:Point = new Point(circle2.x, circle2.y);
pt2 = circle2.localToGlobal(pt2);
var distance:Number = Point.distance(pt1, pt2);
```

Ebenso können Sie den Abstand zwischen dem Registrierungspunkt des Anzeigeobjekts `target` und einem bestimmten Punkt auf der Bühne mithilfe der `localToGlobal()`-Methode der DisplayObject-Klasse ermitteln:

```
import flash.geom.*;
var stageCenter:Point = new Point();
stageCenter.x = this.stage.stageWidth / 2;
stageCenter.y = this.stage.stageHeight / 2;
var targetCenter:Point = new Point(target.x, target.y);
targetCenter = target.localToGlobal(targetCenter);
var distance:Number = Point.distance(stageCenter, targetCenter);
```

Verschieben eines Anzeigeobjekts um einen bestimmten Winkel und Abstand

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der `polar()`-Methode der Point-Klasse können Sie ein Anzeigeobjekt in einem bestimmten Winkel um einen bestimmten Abstand verschieben. Mit dem folgenden Code wird das `myDisplayObject`-Objekt beispielsweise um 100 Pixel im Winkel von 60° verschoben:

```
import flash.geom.*;
var distance:Number = 100;
var angle:Number = 2 * Math.PI * (90 / 360);
var translatePoint:Point = Point.polar(distance, angle);
myDisplayObject.x += translatePoint.x;
myDisplayObject.y += translatePoint.y;
```

Andere Verwendungen der Point-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können Point-Objekte mit den folgenden Methoden und Eigenschaften verwenden:

Klasse	Methoden oder Eigenschaften	Beschreibung
DisplayObjectContainer	`areInaccessibleObjectsUnderPoint()` `getObjectsUnderPoint()`	Wird zum Zurückgeben einer Liste mit Objekten an einem bestimmten Punkt in einem Anzeigeobjektcontainer verwendet.
BitmapData	`hitTest()`	Wird zum Definieren der Pixel im BitmapData-Objekt und des Punktes verwendet, der auf Übereinstimmung geprüft wird.

BitmapData	`applyFilter()`
	`copyChannel()`
	`merge()`
	`paletteMap()`
	`pixelDissolve()`
	`threshold()`
		Wird zum Festlegen der Position von Rechtecken verwendet, mit denen Operationen definiert werden.

Matrix	`deltaTransformPoint()` `transformPoint()`	Wird zum Definieren von Punkten verwendet, auf die eine Transformation angewendet werden soll.

Rectangle	`bottomRight`
	`size`
	`topLeft`
		Wird zum Definieren der Eigenschaften verwendet.

Verwenden von Rectangle-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Rectangle-Objekt definiert einen rechteckigen Bereich. Ein Rectangle-Objekt verfügt über eine Position, die durch die *x*- und *y*-Koordinaten der linken oberen Ecke definiert wird, sowie über `width`- und `height`-Eigenschaften. Sie können diese Eigenschaften für ein neues Rectangle-Objekt definieren, indem Sie die `Rectangle()`-Konstruktorfunktion aufrufen, wie im Folgenden gezeigt:

```
import flash.geom.Rectangle;
var rx:Number = 0;
var ry:Number = 0;
var rwidth:Number = 100;
var rheight:Number = 50;
var rect1:Rectangle = new Rectangle(rx, ry, rwidth, rheight);
```

Ändern der Größe und der Position von Rectangle-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Größe und die Position von Rectangle-Objekten kann auf verschiedene Weise geändert werden.

Sie können die Position des Rectangle-Objekts direkt ändern, indem Sie die zugehörige `x`- und `y`-Eigenschaft ändern. Diese Änderung wirkt sich nicht auf die Breite und Höhe des Rectangle-Objekts aus.

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.x = 20;
rect1.y = 30;
trace(rect1); // (x=20, y=30, w=100, h=50)
```

Wenn Sie die `left`- oder `top`-Eigenschaft eines Rectangle-Objekts ändern, wird das Rechteck neu positioniert, wie im folgenden Code gezeigt. Die `x`- und `y`-Eigenschaften des Rechtecks entsprechen den `left`- und `top`-Eigenschaften. Die Position der linken unteren Ecke des Rectangle-Objekts ändert sich jedoch nicht, daher wird die Größe des Objekts geändert.

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.left = 20;
rect1.top = 30;
trace(rect1); // (x=20, y=30, w=80, h=20)
```

Wenn Sie die `bottom`- oder `right`-Eigenschaft eines Rectangle-Objekts ändern, ändert sich die Position der linken oberen Ecke nicht. Dies wird im folgenden Beispiel gezeigt. Die Größe des Rechtecks wird entsprechend geändert:

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.right = 60;
trect1.bottom = 20;
trace(rect1); // (x=0, y=0, w=60, h=20)
```

Sie können ein Rectangle-Objekt zudem mithilfe der `offset()`-Methode folgendermaßen neu positionieren:

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.offset(20, 30);
trace(rect1); // (x=20, y=30, w=100, h=50)
```

Die Position kann mit der `offsetPt()`-Methode auf ähnliche Weise geändert werden. Bei dieser Methode wird jedoch als Parameter ein Point-Objekt und kein *x*- und *y*-Offset-Wert verwendet.

Sie können die Größe eines Rectangle-Objekts mithilfe der `inflate()`-Methode ändern, die die beiden Parameter `dx` und `dy` enthält. Der `dx`-Parameter gibt die Anzahl der Pixel an, um die die rechte und linke Seite des Rechtecks von der Mitte weg bewegt werden. Der `dy`-Parameter gibt die Anzahl der Pixel an, um die die obere und untere Seite des Rechtecks von der Mitte weg bewegt werden.

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.inflate(6,4);
trace(rect1); // (x=-6, y=-4, w=112, h=58)
```

Die Größe kann mit der `inflatePt()`-Methode auf ähnliche Weise geändert werden. Bei dieser Methode wird jedoch als Parameter ein Point-Objekt und kein `dx`- und `dy`-Wert verwendet.

Vereinigungen und Überschneidungen von Rectangle-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der `union()`-Methode können Sie den rechteckigen Bereich ermitteln, der durch die Begrenzungen zweier Rechtecke gebildet wird:

```
import flash.display.*;
import flash.geom.Rectangle;
var rect1:Rectangle = new Rectangle(0, 0, 100, 100);
trace(rect1); // (x=0, y=0, w=100, h=100)
var rect2:Rectangle = new Rectangle(120, 60, 100, 100);
trace(rect2); // (x=120, y=60, w=100, h=100)
trace(rect1.union(rect2)); // (x=0, y=0, w=220, h=160)
```

Mithilfe der `intersection()`-Methode können Sie den rechteckigen Bereich ermitteln, der durch den sich überschneidenden Bereich zweier Rechtecke gebildet wird:

```
import flash.display.*;
import flash.geom.Rectangle;
var rect1:Rectangle = new Rectangle(0, 0, 100, 100);
trace(rect1); // (x=0, y=0, w=100, h=100)
var rect2:Rectangle = new Rectangle(80, 60, 100, 100);
trace(rect2); // (x=120, y=60, w=100, h=100)
trace(rect1.intersection(rect2)); // (x=80, y=60, w=20, h=40)
```

Mit der `intersects()`-Methode können Sie ermitteln, ob sich zwei Rechtecke überschneiden. Darüber hinaus können Sie mit der `intersects()`-Methode ermitteln, ob sich ein Anzeigeobjekt in einem bestimmten Bereich der Bühne befindet. Im folgenden Codebeispiel wird davon ausgegangen, dass der Koordinatenraum des Anzeigeobjektcontainers, der das `circle`-Objekt enthält, mit dem Koordinatenraum der Bühne identisch ist. Im Beispiel ist dargestellt, wie mithilfe der `intersects()`-Methode festgestellt wird, ob das Anzeigeobjekt `circle` bestimmte Bereiche der Bühne überschneidet, die durch die Rectangle-Objekte `target1` und `target2` definiert sind:

```
import flash.display.*;
import flash.geom.Rectangle;
var circle:Shape = new Shape();
circle.graphics.lineStyle(2, 0xFF0000);
circle.graphics.drawCircle(250, 250, 100);
addChild(circle);
var circleBounds:Rectangle = circle.getBounds(stage);
var target1:Rectangle = new Rectangle(0, 0, 100, 100);
trace(circleBounds.intersects(target1)); // false
var target2:Rectangle = new Rectangle(0, 0, 300, 300);
trace(circleBounds.intersects(target2)); // true
```

Mit der `intersects()`-Methode können Sie ebenso ermitteln, ob sich die Begrenzungsrechtecke zweier Anzeigeobjekte überschneiden. Mit der `getRect()`-Methode der DisplayObject-Klasse können Sie den zusätzlichen Raum erfassen, der einem Begrenzungsbereich durch die Striche eines Anzeigeobjekts hinzugefügt wird.

Andere Verwendungen von Rectangle-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Rectangle-Objekte werden bei den folgenden Methoden und Eigenschaften verwendet:

Klasse	Methoden oder Eigenschaften	Beschreibung
BitmapData	`applyFilter()`, `colorTransform()`, `copyChannel()`, `copyPixels()`, `draw()`, `drawWithQuality()`, `encode()`, `fillRect()`, `generateFilterRect()`, `getColorBoundsRect()`, `getPixels()`, `merge()`, `paletteMap()`, `pixelDissolve()`, `setPixels()` und `threshold()`	Wird als Typ für einige Parameter zum Definieren eines Bereichs des BitmapData-Objekts verwendet.
DisplayObject	`getBounds()`, `getRect()`, `scrollRect`, `scale9Grid`	Wird als Datentyp für die zurückgegebene Eigenschaft oder den zurückgegebenen Datentyp verwendet.
PrintJob	`addPage()`	Wird zum Definieren des `printArea`-Parameters verwendet.
Sprite	`startDrag()`	Wird zum Definieren des `bounds`-Parameters verwendet.
TextField	`getCharBoundaries()`	Wird als Rückgabewerttyp verwendet.
Transform	`pixelBounds`	Wird als Datentyp verwendet.

Verwenden von Matrix-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Matrix-Klasse repräsentiert eine Transformationsmatrix, die bestimmt, wie Punkte zwischen verschiedenen Koordinatenräumen abgebildet werden. Sie können verschiedene grafische Transformationen für ein Anzeigeobjekt durchführen, indem Sie die Eigenschaften eines „Matrix“-Objekts festlegen, dieses „Matrix“-Objekt auf die `matrix`-Eigenschaft eines „Transform“-Objekts anwenden und dieses „Transform“-Objekt dann als `transform`-Eigenschaft des Anzeigeobjekts anwenden. Die verfügbaren Transformationsfunktionen sind Versetzen (*x*- und *y*-Neupositionierung), Drehen, Skalieren und Neigen.

Obwohl eine Matrix definiert werden kann, indem die Eigenschaften (`a`, `b`, `c`, `d`, `tx`, `ty`) eines Matrix-Objekts direkt festgelegt werden, ist es einfacher, die `createBox()`-Methode zu verwenden. Diese Methode enthält Parameter, über die Sie die Effekte zum Skalieren, Drehen und Versetzen der resultierenden Matrix direkt definieren können. Mit dem folgenden Code wird beispielsweise ein Matrix-Objekt erstellt, das ein Objekt horizontal mit dem Faktor 2,0 und vertikal mit dem Faktor 3,0 skaliert, um 45 Grad dreht sowie um 10 Pixel nach rechts und um 20 Pixel nach unten verschiebt (versetzt):

```
var matrix:Matrix = new Matrix();
var scaleX:Number = 2.0;
var scaleY:Number = 3.0;
var rotation:Number = 2 * Math.PI * (45 / 360);
var tx:Number = 10;
var ty:Number = 20;
matrix.createBox(scaleX, scaleY, rotation, tx, ty);
```

Sie können die Effekte eines Matrix-Objekts zum Skalieren, Drehen und Versetzen auch mithilfe der Methoden `scale()`, `rotate()` und `translate()` anpassen. Beachten Sie, dass bei diesen Methoden die übergebenen Parameter mit den Werten des bestehenden Matrix-Objekts kombiniert werden. Mit dem folgenden Code wird beispielsweise ein Matrix-Objekt festgelegt, mit dem ein Objekt um den Faktor 4 skaliert und um 60° gedreht wird, da die Methoden `scale()` und `rotate()` zweimal aufgerufen werden:

```
var matrix:Matrix = new Matrix();
var rotation:Number = 2 * Math.PI * (30 / 360); // 30°
var scaleFactor:Number = 2;
matrix.scale(scaleFactor, scaleFactor);
matrix.rotate(rotation);
matrix.scale(scaleX, scaleY);
matrix.rotate(rotation);

myDisplayObject.transform.matrix = matrix;
```

Wenn Sie ein Matrix-Objekt neigen möchten, ändern Sie die entsprechende `b`- oder `c`-Eigenschaft. Durch Ändern der `b`-Eigenschaft wird die Matrix vertikal geneigt und durch Ändern der `c`-Eigenschaft wird sie horizontal geneigt. Mit dem folgenden Code wird das Matrix-Objekt `myMatrix` um den Faktor 2 vertikal geneigt:

```
var skewMatrix:Matrix = new Matrix();
skewMatrix.b = Math.tan(2);
myMatrix.concat(skewMatrix);
```

Sie können eine Matrixtransformation auf die `transform`-Eigenschaft eines Anzeigeobjekts anwenden. Mit dem folgenden Code wird beispielsweise eine Matrixtransformation auf das Anzeigeobjekt `myDisplayObject` angewendet:

```
var matrix:Matrix = myDisplayObject.transform.matrix;
var scaleFactor:Number = 2;
var rotation:Number = 2 * Math.PI * (60 / 360); // 60°
matrix.scale(scaleFactor, scaleFactor);
matrix.rotate(rotation);

myDisplayObject.transform.matrix = matrix;
```

In der ersten Zeile wird ein Matrix-Objekt auf die vorhandene Transformationsmatrix gesetzt, die im `myDisplayObject`-Anzeigeobjekt verwendet wird (die `matrix`-Eigenschaft der `transformation`-Eigenschaft des `myDisplayObject` -Anzeigeobjekts). Auf diese Weise haben die Methoden der aufgerufenen Matrix-Klasse eine kumulative Auswirkung auf die aktuelle Position, Skalierung und Drehung des Anzeigeobjekts.

Hinweis: *Das flash.geometry-Paket enthält auch die ColorTransform-Klasse. Mit dieser Klasse wird die* `colorTransform`*-Eigenschaft eines Transform-Objekts festgelegt. Da hierbei keine geometrischen Transformationen angewendet werden, wird sie hier nicht näher erläutert. Weitere Informationen finden Sie in der Beschreibung der ColorTransform-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.*

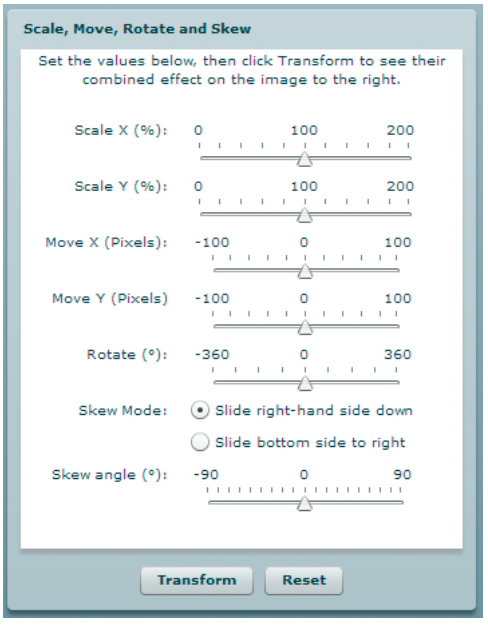
Geometrie-Beispiel: Anwenden einer Matrixtransformation auf ein Anzeigeobjekt

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Beispielanwendung „DisplayObjectTransformer“ veranschaulicht mehrere Funktionen, über die ein Anzeigeobjekt mithilfe der Matrix-Klasse geändert werden kann. Dazu gehören:

- Drehen des Anzeigeobjekts
- Skalieren des Anzeigeobjekts
- Versetzen (Neupositionieren) des Anzeigeobjekts
- Neigen des Anzeigeobjekts

Die Anwendung stellt die folgende Benutzeroberfläche zum Anpassen der Parameter der Matrixtransformation bereit:



Wenn der Benutzer auf die Schaltfläche „Transformieren“ klickt, wird die entsprechende Transformation angewendet.

Das ursprüngliche Anzeigeobjekt und das um -45 ° gedrehte und um 50 % skalierte Anzeigeobjekt

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „DisplayObjectTransformer“ befinden sich im Ordner „Samples/DisplayObjectTransformer“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
DisplayObjectTransformer.mxml oder DisplayObjectTransformer.fla	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/geometry/MatrixTransformer.as	Eine Klasse mit Methoden zum Anwenden von Matrixtransformationen.
img/	Ein Verzeichnis mit den in der Anwendung verwendeten Beispielbilddateien.

Definieren der MatrixTransformer-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die MatrixTransformer-Klasse enthält statische Methoden, die geometrische Transformationen von Matrix-Objekten anwenden.

transform()-Methode

Die `transform()`-Methode enthält folgende Parameter:

- `sourceMatrix` – Die Eingabematrix, die mit der Methode transformiert wird.
- `xScale` und `yScale` – Der Skalierungsfaktor für *x* und *y*.
- `dx` und `dy` – Die Versetzung von *x* und *y* in Pixel.
- `rotation` – Die Drehung in Grad.
- `skew` – Der Neigungsfaktor als Prozentsatz.
- `skewType` – Die Neigungsrichtung, entweder `right` oder `left`.

Der Rückgabewert ist die resultierende Matrix.

Mit der `transform()`-Methode werden die folgenden statischen Methoden der Klasse aufgerufen:

- `skew()`
- `scale()`

- `translate()`
- `rotate()`

Jede Methode gibt die Quellmatrix mit der angewendeten Transformation zurück.

skew()-Methode

Mit der `skew()`-Methode wird die Matrix geneigt, indem die Eigenschaften `b` und `c` der Matrix angepasst werden. Mit dem optionalen Parameter `unit` werden die Einheiten ermittelt, die zum Definieren des Neigungswinkels verwendet werden. Gegebenenfalls konvertiert die Methode den `angle`-Wert in das Bogenmaß:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
```

Das Matrix-Objekt `skewMatrix` wird erstellt und so angepasst, dass die Neigungstransformation angewendet werden kann. Anfangs ist dies die Identitätsmatrix, wie im Folgenden dargestellt:

```
var skewMatrix:Matrix = new Matrix();
```

Der Parameter `skewSide` gibt die Seite an, in deren Richtung die Neigung angewendet wird. Wenn der Parameter auf `right` gesetzt ist, wird mit dem folgenden Code die `b`-Eigenschaft der Matrix festgelegt:

```
skewMatrix.b = Math.tan(angle);
```

Andernfalls wird die untere Seite geneigt, indem die `c`-Eigenschaft der Matrix angepasst wird, wie im Folgenden dargestellt:

```
skewMatrix.c = Math.tan(angle);
```

Die resultierende Neigung wird dann auf die vorhandene Matrix angewendet, indem die beiden Matrizen verkettet werden, wie im folgenden Beispiel dargestellt:

```
sourceMatrix.concat(skewMatrix);
return sourceMatrix;
```

scale()-Methode

Wie im folgenden Beispiel dargestellt, wird mit der `scale()`-Methode zunächst der Skalierungsfaktor angepasst, wenn dieser als Prozentsatz angegeben ist, und dann die `scale()`-Methode des Matrix-Objekts verwendet:

```
if (percent)
{
    xScale = xScale / 100;
    yScale = yScale / 100;
}
sourceMatrix.scale(xScale, yScale);
return sourceMatrix;
```

translate()-Methode

Die `translate()`-Methode wendet die Versetzungsfaktoren `dx` und `dy` an, indem die `translate()`-Methode des Matrix-Objekts aufgerufen wird, wie im Folgenden dargestellt:

```
sourceMatrix.translate(dx, dy);
return sourceMatrix;
```

rotate()-Methode

Die `rotate()`-Methode konvertiert den eingegebenen Drehungsfaktor in das Bogenmaß (wenn der Faktor in Grad oder Gradient angegeben wird) und ruft dann die `rotate()`-Methode des Matrix-Objekts auf:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
sourceMatrix.rotate(angle);
return sourceMatrix;
```

Aufrufen der MatrixTransformer.transform()-Methode aus der Anwendung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Anwendung verfügt über eine Benutzeroberfläche zum Abfragen der Transformationsparameter vom Benutzer. Diese Werte werden dann zusammen mit der `matrix`-Eigenschaft der `transform`-Eigenschaft des Anzeigeobjekts wie folgt an die `Matrix.transform()`-Methode übergeben:

```
tempMatrix = MatrixTransformer.transform(tempMatrix,
    xScaleSlider.value,
    yScaleSlider.value,
    dxSlider.value,
    dySlider.value,
    rotationSlider.value,
    skewSlider.value,
    skewSide );
```

Anschließend wird der Rückgabewert auf die `matrix`-Eigenschaft der `transform`-Eigenschaft des Anzeigeobjekts angewendet. Dadurch wird die Transformation ausgelöst:

```
img.content.transform.matrix = tempMatrix;
```

Kapitel 12: Verwenden der Zeichnungs-API

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Obwohl importierte Bilder und Vorlagen wichtig sind, haben Sie mithilfe der sogenannten Zeichnungs-API, mit der Linien und Formen in ActionScript gezeichnet werden können, die Möglichkeit, eine Anwendung sozusagen mit einer leeren Leinwand zu starten, auf der Sie alle gewünschten Bilder selbst erstellen können. Diese Funktion zum Erstellen eigener Grafiken eröffnet weitreichende Möglichkeiten für Ihre Anwendungen. Mit den hier erörterten Verfahren können Sie unter anderem ein Zeichenprogramm, animierte und interaktive Bilder oder programmgesteuert benutzerdefinierte Elemente auf der Benutzeroberfläche erstellen.

Verwandte Hilfethemen

flash.display.Graphics

Grundlagen der Zeichnungs-API

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der integrierte Funktionsumfang von ActionScript, mit dem Sie Vektorgrafiken, d. h. Linien, Kurven, Formen, Füllungen und Farbverläufe, erstellen und mit ActionScript auf dem Bildschirm anzeigen können, wird als Zeichnungs-API bezeichnet. Diese Funktionalität wird von der flash.display.Graphics-Klasse bereitgestellt. Mit ActionScript können Sie in allen Instanzen der Klassen Shape, Sprite oder MovieClip zeichnen, indem Sie die in jeder dieser Klassen definierte `graphics`-Eigenschaft verwenden. (Bei der `graphics`-Eigenschaft dieser Klassen handelt es sich jeweils um eine Instanz der Graphics-Klasse.)

Für den Fall, dass Sie gerade erst mit dem programmgesteuerten Zeichnen beginnen, enthält die Graphics-Klasse mehrere Methoden, die das Zeichnen geläufiger Formen wie Kreise, Ellipsen, Rechtecke und Rechtecke mit abgerundeten Ecken erleichtern. Sie können diese Formen mit Rahmenlinien oder als gefüllte Formen zeichnen. Für einen erweiterten Funktionsumfang enthält die Graphics-Klasse zudem Methoden zum Zeichnen von Linien und quadratischen Bézier-Kurven, die Sie mit den Trigonometriefunktionen der Math-Klasse verwenden können, um die gewünschten Formen zu erstellen.

Flash-Laufzeitumgebungen (wie Flash Player 10 und Adobe AIR 1.5 und höher) fügen eine weitere API zum Zeichnen hinzu, die Ihnen das programmgesteuerte Zeichnen ganzer Formen mit einem einzigen Befehl ermöglicht. Nachdem Sie sich mit der Graphics-Klasse und den unter „Grundlagen der Verwendung der Zeichnungs-API“ beschriebenen Aufgaben vertraut gemacht haben, lesen Sie „Erweiterte Einsatzmöglichkeiten der Zeichnungs-API“ auf Seite 250, um sich über diese Funktionen der Zeichnungs-API zu informieren.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe aufgeführt, die Ihnen beim Verwenden der Zeichnungs-API begegnen:

Ankerpunkt Einer der beiden Endpunkte einer quadratischen Bézier-Kurve.

Steuerpunkt Der Punkt, mit dem die Richtung und Krümmung einer quadratischen Bézier-Kurve definiert wird. Die gekrümmte Linie erreicht nie den Kontrollpunkt. Sie wird jedoch gekrümmt, als ob sie in Richtung des Kontrollpunkts gezogen wird.

Koordinatenraum Das Koordinatendiagramm eines Anzeigeobjekts, in dem die jeweils untergeordneten Elemente positioniert sind.

Füllung Der gefüllte innere Teil einer Form mit einer mit Farbe gefüllten Linie oder eine gesamte Form ohne Kontur.

Gradient Eine Farbe, bei der eine Farbe graduell in eine oder mehrere andere Farben übergeht (im Gegensatz zu einer Volltonfarbe).

Punkt Eine einzelne Position in einem Koordinatenraum. In dem in ActionScript verwendeten zweidimensionalen Koordinatensystem wird ein Punkt durch seine Position auf der x-Achse und der y-Achse (die Koordinaten des Punkts) definiert.

Quadratische Bézier-Kurve Eine Kurve, die durch eine bestimmte mathematische Formel definiert ist. Der Verlauf der Kurve wird anhand der Position der Ankerpunkte (der Endpunkte der Kurve) und anhand eines Kontrollpunkts berechnet, der die Krümmung und Richtung der Kurve definiert.

Skalieren Die Größe eines Objekts im Verhältnis zu seiner Originalgröße. Beim Skalieren eines Objekts wird die Größe des Objekts geändert, indem das Objekt vergrößert oder verkleinert wird.

Strich Die Kontur einer Form mit einer mit Farbe gefüllten Linie oder die Linien einer nicht gefüllten Form.

Versetzen Umwandeln der Koordinaten eines Punkts von einem Koordinatenraum in einen anderen Koordinatenraum.

X-Achse Die horizontale Achse des in ActionScript verwendeten zweidimensionalen Koordinatensystems.

Y-Achse Die vertikale Achse des in ActionScript verwendeten zweidimensionalen Koordinatensystems.

Die Graphics-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jedes Shape-, Sprite- und MovieClip-Objekt enthält eine `graphics`-Eigenschaft, bei der es sich um eine Instanz der Graphics-Klasse handelt. Die Graphics-Klasse enthält Eigenschaften und Methoden zum Zeichnen von Linien, Füllungen und Formen. Wenn Sie ein Anzeigeobjekt lediglich als Leinwand zum Zeichnen von Inhalten verwenden möchten, können Sie dazu eine Shape-Instanz einsetzen. Eine Shape-Instanz eignet sich besser als andere Anzeigeobjekte zum Zeichnen, da sie nicht über die zusätzlichen Funktionen der Sprite-Klasse oder der MovieClip-Klasse verfügt. Wenn Sie ein Anzeigeobjekt verwenden möchten, auf dem Sie grafische Inhalte zeichnen können und das andere Anzeigeobjekte enthalten kann, können Sie dazu eine Sprite-Instanz einsetzen. Weitere Informationen zur Verwendung der verschiedenen Anzeigeobjekte für die unterschiedlichen Aufgaben finden Sie unter „Auswählen einer DisplayObject-Unterklasse“ auf Seite 184.

Zeichnen von Linien und Kurven

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Alle Zeichnungen mithilfe einer Graphics-Instanz beruhen auf dem Zeichnen von Linien und Kurven. Folglich müssen alle Zeichenvorgänge in ActionScript mit denselben Schritten durchgeführt werden:

- Definieren von Linien- und Füllstilen
- Festlegen der anfänglichen Zeichnungsposition
- Zeichnen von Linien, Kurven und Formen (optional Verschieben des Zeichnungspunkts)
- Gegebenenfalls Abschließen einer Füllung

Definieren von Linien- und Füllstilen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um mithilfe der `graphics`-Eigenschaft einer Shape-, Sprite- oder MovieClip-Instanz zeichnen zu können, müssen Sie zunächst den Stil (Liniengröße und -farbe, Füllfarbe) festlegen, der beim Zeichnen verwendet wird. Wie bei der Verwendung der Zeichenwerkzeuge in Adobe® Flash® Professional oder einer anderen Zeichenanwendung können Sie beim Zeichnen mit ActionScript Zeichnungen mit oder ohne Strich sowie mit oder ohne Füllfarbe erstellen. Sie können das Erscheinungsbild der Striche mit der `lineStyle()`-Methode oder der `lineGradientStyle()`-Methode angeben. Mit der `lineStyle()`-Methode können Sie gefüllte Linien erstellen. Beim Aufrufen dieser Methode übergeben Sie in den meisten Fällen Werte für die ersten drei Parameter: Linienstärke, Farbe und Alpha. Mit der folgenden Codezeile werden beispielsweise mit dem Shape-Objekt `myShape` Linien gezeichnet, die eine Linienstärke von 2 Pixel, die Farbe Rot (0x990000) und eine Opazität von 75 % aufweisen:

```
myShape.graphics.lineStyle(2, 0x990000, .75);
```

Der Standardwert für den Alphaparameter ist 1,0 (100 %), daher müssen Sie diesen Parameter nicht angeben, wenn Sie eine vollständig undurchsichtige Linie erstellen möchten. Bei der `lineStyle()`-Methode können Sie zwei weitere Parameter für die Anzeige von Strichen als ganze Pixel und für den Skalierungsmodus angeben. Weitere Informationen zur Verwendung dieser Parameter finden Sie in der Beschreibung der `Graphics.lineStyle()`-Methode im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Mit der `lineGradientStyle()`-Methode können Sie Farbverlaufslinien erstellen. Diese Methode wird unter „Erstellen von Farbverlaufslinien und -füllungen“ auf Seite 241 beschrieben.

Wenn Sie eine gefüllte Form erstellen möchten, rufen Sie zunächst die Methode `beginFill()`, `beginGradientFill()` oder `beginBitmapFill()` oder `beginShaderFill()` auf und starten dann die Zeichnung. Die Basismethode `beginFill()` akzeptiert zwei Parameter: die Füllfarbe und (optional) einen Alphawert für die Füllfarbe. Wenn Sie z. B. eine Form mit einer einfarbigen grünen Füllung zeichnen möchten, verwenden Sie den folgenden Code (für ein Objekt mit dem Namen `myShape`):

```
myShape.graphics.beginFill(0x00FF00);
```

Durch Aufruf einer Füllmethode wird implizit eine bisherige Füllung beendet, bevor eine neue Füllung begonnen wird. Beim Aufrufen einer Methode, mit der ein Strichstil angegeben wird, wird der vorherige Strich ersetzt, während eine zuvor angegebene Füllung nicht geändert wird, und umgekehrt.

Nach dem Angeben des Linienstils und der Fülleigenschaften geben Sie als Nächstes den Anfangspunkt der Zeichnung an. Die Graphics-Instanz verfügt über einen Zeichnungspunkt, ähnlich der Zeichenstiftspitze auf dem Papier. Der jeweils nächste Zeichenvorgang beginnt an der entsprechenden Position des Zeichnungspunkts. In der Standardeinstellung befindet sich der Zeichnungspunkt eines Graphics-Objekts an der Position (0,0) des Koordinatenraums des gezeichneten Objekts. Wenn Sie die Zeichnung an einer anderen Position beginnen möchten, rufen Sie zunächst die `moveTo()`-Methode und anschließend eine der Zeichnungsmethoden auf. Dies entspricht dem Bewegen der Zeichenstiftspitze an eine andere Stelle auf dem Papier.

Ausgehend vom Zeichnungspunkt erstellen Sie Zeichnungen durch mehrere Aufrufe der Zeichnungsmethoden `lineTo()` (Zeichnen gerader Linien) und `curveTo()` (Zeichnen gekrümmter Linien).

Beim Zeichnen können Sie jederzeit die `moveTo()`-Methode aufrufen, um den Zeichnungspunkt an eine neue Position zu verschieben, ohne dabei zu zeichnen.

Wenn Sie eine Füllfarbe angegeben haben, können Sie beim Zeichnen die Füllung abschließen, indem Sie die `endFill()`-Methode aufrufen. Wenn Sie keine geschlossene Form gezeichnet haben (d. h., wenn sich der Zeichnungspunkt beim Aufrufen von `endFill()` nicht am Anfangspunkt der Form befindet) und dann die `endFill()`-Methode aufrufen, wird die Form in der Flash-Laufzeitumgebung automatisch geschlossen, indem eine gerade Linie vom aktuellen Zeichnungspunkt zu der Position gezogen wird, die beim letzten Aufruf von `moveTo()` angegeben wurde. Wenn Sie eine Füllung begonnen und `endFill()` nicht aufgerufen haben, wird durch Aufruf von `beginFill()` (oder einer der anderen Füllmethoden) die aktuelle Füllung beendet und eine neue Füllung begonnen.

Zeichnen gerader Linien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Durch Aufrufen der `lineTo()`-Methode wird mit dem Graphics-Objekt eine gerade Linie vom aktuellen Zeichnungspunkt zu den Koordinaten gezogen, die Sie als beide Parameter beim Aufruf der Methode angeben, und zwar mit dem angegebenen Linienstil. In der folgenden Codezeile wird der Zeichnungspunkt beispielsweise an die Position (100, 100) verschoben. Anschließend wird eine Linie zur Position (200, 200) gezogen:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.lineTo(200, 200);
```

Im folgenden Beispiel werden rote und grüne Dreiecke mit einer Höhe von 100 Pixel erstellt:

```
var triangleHeight:uint = 100;
var triangle:Shape = new Shape();

// red triangle, starting at point 0, 0
triangle.graphics.beginFill(0xFF0000);
triangle.graphics.moveTo(triangleHeight / 2, 0);
triangle.graphics.lineTo(triangleHeight, triangleHeight);
triangle.graphics.lineTo(0, triangleHeight);
triangle.graphics.lineTo(triangleHeight / 2, 0);

// green triangle, starting at point 200, 0
triangle.graphics.beginFill(0x00FF00);
triangle.graphics.moveTo(200 + triangleHeight / 2, 0);
triangle.graphics.lineTo(200 + triangleHeight, triangleHeight);
triangle.graphics.lineTo(200, triangleHeight);
triangle.graphics.lineTo(200 + triangleHeight / 2, 0);

this.addChild(triangle);
```

Zeichnen von Kurven

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `curveTo()`-Methode wird eine quadratische Bézier-Kurve gezeichnet. Dabei wird ein Bogen erstellt, der zwei Punkte (die sogenannten Ankerpunkte) verbindet und sich gleichzeitig in Richtung eines dritten Punktes (dem sogenannten Kontrollpunkt) spannt. Im Graphics-Objekt dient die aktuelle Zeichnungsposition als erster Ankerpunkt. Beim Aufrufen der `curveTo()`-Methode werden vier Parameter übergeben: die x- und y-Koordinate des Kontrollpunkts, gefolgt von der x- und y-Koordinate des zweiten Ankerpunkts. Mit dem folgenden Code wird beispielsweise eine Kurve mit dem Anfangspunkt (100, 100) und dem Endpunkt (200, 200) gezeichnet. Da sich der Kontrollpunkt an Position (175, 125) befindet, wird eine Kurve erstellt, die nach rechts und dann nach unten gekrümmt ist:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.curveTo(175, 125, 200, 200);
```

Im folgenden Beispiel werden rote und grüne kreisförmige Objekte mit einer Breite und Höhe von 100 Pixel erstellt. Beachten Sie, dass es sich aufgrund der Merkmale der quadratischen Bézier-Gleichung nicht um perfekt geformte Kreise handelt:

```
var size:uint = 100;
var roundObject:Shape = new Shape();

// red circular shape
roundObject.graphics.beginFill(0xFF0000);
roundObject.graphics.moveTo(size / 2, 0);
roundObject.graphics.curveTo(size, 0, size, size / 2);
roundObject.graphics.curveTo(size, size, size / 2, size);
roundObject.graphics.curveTo(0, size, 0, size / 2);
roundObject.graphics.curveTo(0, 0, size / 2, 0);

// green circular shape
roundObject.graphics.beginFill(0x00FF00);
roundObject.graphics.moveTo(200 + size / 2, 0);
roundObject.graphics.curveTo(200 + size, 0, 200 + size, size / 2);
roundObject.graphics.curveTo(200 + size, size, 200 + size / 2, size);
roundObject.graphics.curveTo(200, size, 200, size / 2);
roundObject.graphics.curveTo(200, 0, 200 + size / 2, 0);

this.addChild(roundObject);
```

Zeichnen von Formen mit integrierten Methoden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für das Zeichnen von häufig vorkommenden Formen wie Kreise, Ellipsen, Rechtecke und Rechtecke mit abgerundeten Ecken enthält ActionScript 3.0 zur leichteren Handhabung entsprechende Methoden. Dabei handelt es sich um die Methoden `drawCircle()`, `drawEllipse()`, `drawRect()` und `drawRoundRect()` der Graphics-Klasse. Diese Methoden können anstelle der Methoden `lineTo()` und `curveTo()` verwendet werden. Sie müssen dennoch vor dem Aufrufen dieser Methoden Linien- und Füllstile angeben.

Im folgenden Beispiel wird das Codebeispiel zum Erstellen roter, grüner und blauer Quadrate mit einer Breite und Höhe von 100 Pixel erneut verwendet. In diesem Codebeispiel kommt die `drawRect()`-Methode zum Einsatz. Zusätzlich wird für die Füllfarbe der Alphawert 50 % (0,5) angegeben:

```
var squareSize:uint = 100;
var square:Shape = new Shape();
square.graphics.beginFill(0xFF0000, 0.5);
square.graphics.drawRect(0, 0, squareSize, squareSize);
square.graphics.beginFill(0x00FF00, 0.5);
square.graphics.drawRect(200, 0, squareSize, squareSize);
square.graphics.beginFill(0x0000FF, 0.5);
square.graphics.drawRect(400, 0, squareSize, squareSize);
square.graphics.endFill();
this.addChild(square);
```

In einem Sprite- oder MovieClip-Objekt wird der mit der `graphics`-Eigenschaft erstellte Zeicheninhalt immer hinter allen untergeordneten Anzeigeobjekten des jeweiligen Objekts angezeigt. Darüber hinaus handelt es sich beim Inhalt der `graphics`-Eigenschaft um kein eigenständiges Anzeigeobjekt, sodass es nicht in der Liste der untergeordneten Objekte eines Sprite- oder MovieClip-Objekts angezeigt wird. Um das folgende Sprite-Objekt wird beispielsweise mit der `graphics`-Eigenschaft ein Kreis gezogen. Zudem wird ein TextField-Objekt in der Liste der zugehörigen untergeordneten Anzeigeobjekte angezeigt:

```
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0xFFCC00);
mySprite.graphics.drawCircle(30, 30, 30);
var label:TextField = new TextField();
label.width = 200;
label.text = "They call me mellow yellow...";
label.x = 20;
label.y = 20;
mySprite.addChild(label);
this.addChild(mySprite);
```

Beachten Sie, dass das TextField-Objekt oberhalb des mit dem Graphics-Objekts gezeichneten Kreises angezeigt wird.

Erstellen von Farbverlaufslinien und -füllungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem Graphics-Objekt können zudem Striche und Füllungen mit Farbverläufen anstelle von Volltonfarben gezeichnet werden. Ein Farbverlaufsstrich wird mit der `lineGradientStyle()`-Methode erstellt und eine Farbverlaufsfüllung mit der `beginGradientFill()`-Methode.

Bei beiden Methoden werden dieselben Parameter angegeben. Die ersten vier sind erforderlich: Typ, Farben, Alphawerte und Verhältnisse. Die übrigen vier Parameter sind optional, jedoch nützlich zur erweiterten Anpassung.

- Mit dem ersten Parameter wird der Typ des zu erstellenden Farbverlaufs angegeben. Zulässige Werte sind `GradientType.LINEAR` oder `GradientType.RADIAL`.
- Mit dem zweiten Parameter wird das Array der zu verwendenden Farben angegeben. Bei einem linearen Farbverlauf sind die Farben von links nach rechts angeordnet. Bei einem radialen Farbverlauf sind die Farben von innen nach außen angeordnet. Die Anordnung der Farben im Array stellt die Reihenfolge dar, in der die Farben im Farbverlauf angezeigt werden.

- Mit dem dritten Parameter werden die Werte der Alphatransparenz der entsprechenden Farben im vorherigen Parameter angegeben.
- Mit dem vierten Parameter werden die Verhältnisse bzw. die Gewichtung der einzelnen Farben im Farbverlauf festgelegt. Dabei können Werte in einem Bereich zwischen 0 und 255 angegeben werden. Diese Werte stellen keine Breite oder Höhe dar, sondern die Position im Farbverlauf. Mit 0 wird der Beginn des Farbverlaufs und mit 255 das Ende des Farbverlaufs angegeben. Das Array der Verhältnisse muss sequenziell ansteigen und über die gleiche Anzahl Einträge wie in den Arrays der Farben und Alphawerte verfügen, die im zweiten und dritten Parameter angegeben sind.

Obwohl es sich beim fünften Parameter für die Transformationsmatrix um einen optionalen Parameter handelt, wird dieser Parameter in der Regel verwendet, da so die Darstellung des Farbverlaufs einfach und effizient gesteuert werden kann. Für diesen Parameter kann eine Matrix-Instanz angegeben werden. Die einfachste Möglichkeit, ein Matrix-Objekt für einen Farbverlauf zu erstellen, besteht darin, die `createGradientBox()`-Methode der Matrix-Klasse zu verwenden.

Definieren eines Matrix-Objekts für einen Farbverlauf

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der Methoden `beginGradientFill()` und `lineGradientStyle()` der flash.display.Graphics-Klasse können Sie den Farbverlauf in Formen definieren. Beim Definieren eines Farbverlaufs geben Sie eine Matrix als einen der Parameter dieser Methoden an.

Die Matrix kann am einfachsten mithilfe der `createGradientBox()`-Methode der Matrix-Klasse definiert werden, bei der eine Matrix zum Definieren des Farbverlaufs festgelegt wird. Sie legen die Skalierung, Drehung und Position des Farbverlaufs mit den Parametern fest, die an die `createGradientBox()`-Methode übergeben werden. Bei der `createGradientBox()`-Methode können folgende Parameter angegeben werden:

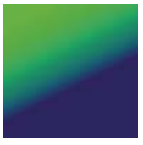
- Breite des Farbverlaufsfelds: die Breite (in Pixel), auf die sich der Farbverlauf ausdehnt
- Höhe des Farbverlaufsfelds: die Höhe (in Pixel), auf die sich der Farbverlauf ausdehnt
- Drehung des Farbverlaufsfelds: die Drehung (in Bogenmaß) des Farbverlaufs
- Horizontale Versetzung: horizontale Versetzung (in Pixel) des Farbverlaufs
- Vertikale Versetzung: vertikale Versetzung (in Pixel) des Farbverlaufs

Dies wird an einem Beispiel für einen Farbverlauf mit den folgenden Merkmalen verdeutlicht:

- `GradientType.LINEAR`
- Zwei Farben, Grün und Blau, bei denen das `ratios`-Array auf `[0, 255]` gesetzt ist.
- `SpreadMethod.PAD`
- `InterpolationMethod.LINEAR_RGB`

In den folgenden Beispielen sind Farbverläufe abgebildet, bei denen der `rotation`-Parameter der `createGradientBox()`-Methode wie angegeben abweicht, alle anderen Einstellungen jedoch unverändert bleiben:

`width = 100;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`	

`width = 100;`
`height = 100;`
`rotation = Math.PI/4; // 45°`
`tx = 0;`
`ty = 0;`
	

`width = 100;`
`height = 100;`
`rotation = Math.PI/2; // 90°`
`tx = 0;`
`ty = 0;`	

In den folgenden Beispielen sind die Effekte in einem linearen Grün-Blau-Farbverlauf abgebildet, bei dem die Parameter `rotation`, `tx` und `ty` der `createGradientBox()`-Methode wie angegeben abweichen, alle anderen Einstellungen jedoch unverändert bleiben:

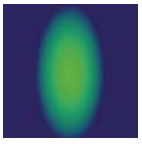
`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`	

`width = 50;`
`height = 100;`
`rotation = 0`
`tx = 50;`
`ty = 0;`	

`width = 100;`
`height = 50;`
`rotation = Math.PI/2; // 90°`
`tx = 0;`
`ty = 0;`	

`width = 100;`
`height = 50;`
`rotation = Math.PI/2; // 90°`
`tx = 0;`
`ty = 50;`	

Die Parameter `width`, `height`, `tx` und `ty` der `createGradientBox()`-Methode wirken sich auch auf die Größe und die Position der *radialen* Farbverlaufsfüllung aus, wie im folgenden Beispiel abgebildet:

`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 25;`
`ty = 0;`
	

Der zuletzt abgebildete radiale Farbverlauf wird mit folgendem Code erstellt:

```
import flash.display.Shape;
import flash.display.GradientType;
import flash.geom.Matrix;

var type:String = GradientType.RADIAL;
var colors:Array = [0x00FF00, 0x000088];
var alphas:Array = [1, 1];
var ratios:Array = [0, 255];
var spreadMethod:String = SpreadMethod.PAD;
var interp:String = InterpolationMethod.LINEAR_RGB;
var focalPtRatio:Number = 0;

var matrix:Matrix = new Matrix();
var boxWidth:Number = 50;
var boxHeight:Number = 100;
var boxRotation:Number = Math.PI/2; // 90°
var tx:Number = 25;
var ty:Number = 0;
matrix.createGradientBox(boxWidth, boxHeight, boxRotation, tx, ty);

var square:Shape = new Shape;
square.graphics.beginGradientFill(type,
                           colors,
                           alphas,
                           ratios,
                           matrix,
                           spreadMethod,
                           interp,
                           focalPtRatio);
square.graphics.drawRect(0, 0, 100, 100);
addChild(square);
```

Beachten Sie, dass die Breite und die Höhe der Farbverlaufsfüllung durch die Breite und die Höhe der Farbverlaufsmatrix und nicht durch die mit dem Graphics-Objekt gezeichnete Breite und Höhe festgelegt werden. Beim Zeichnen mithilfe des Graphics-Objekts wird eine Zeichnung an den entsprechenden Koordinaten in der Farbverlaufsmatrix erstellt. Auch wenn Sie eine der Shape-Methoden eines Graphics-Objekts (z. B. `drawRect()` verwenden, dehnt sich der Farbverlauf nicht automatisch auf die gezeichnete Form aus. Die Größe des Farbverlaufsfelds muss stattdessen in der Farbverlaufsmatrix angegeben werden.

Das folgende Codebeispiel veranschaulicht den visuellen Unterschied zwischen den Abmessungen der Farbverlaufsmatrix und den Zeichnungsabmessungen:

```
var myShape:Shape = new Shape();
var gradientBoxMatrix:Matrix = new Matrix();
gradientBoxMatrix.createGradientBox(100, 40, 0, 0, 0);
myShape.graphics.beginGradientFill(GradientType.LINEAR, [0xFF0000, 0x00FF00, 0x0000FF], [1,
1, 1], [0, 128, 255], gradientBoxMatrix);
myShape.graphics.drawRect(0, 0, 50, 40);
myShape.graphics.drawRect(0, 50, 100, 40);
myShape.graphics.drawRect(0, 100, 150, 40);
myShape.graphics.endFill();
this.addChild(myShape);
```

Mit diesem Code werden drei Farbverläufe mit dem gleichen Füllstil gezeichnet, der als gleichmäßige Verteilung auf die Farben Rot, Grün und Blau angegeben ist. Die Farbverläufe werden mit der `drawRect()`-Methode mit einer Breite von jeweils 50, 100 und 150 Pixel erstellt. Die Farbverlaufsmatrix, die in der `beginGradientFill()`-Methode angegeben ist, wird mit einer Breite von 100 Pixel erstellt. Dies bedeutet, dass der erste Farbverlauf nur die Hälfte des Farbverlaufsspektrums umfasst. Der zweite Farbverlauf umfasst das gesamte Spektrum. Der dritte Farbverlauf umfasst das gesamte Spektrum und weist eine zusätzliche Ausdehnung der Farbe Blau um 50 Pixel nach rechts auf.

Die `lineGradientStyle()`-Methode entspricht der `beginGradientFill()`-Methode, mit der Ausnahme, dass Sie neben der Festlegung des Farbverlaufs vor dem Zeichnen auch die Strichstärke mit der `lineStyle()`-Methode angeben müssen. Mit dem folgenden Code wird ein Feld mit einem Rot-Grün-Blau-Farbverlaufsstrich erstellt:

```
var myShape:Shape = new Shape();
var gradientBoxMatrix:Matrix = new Matrix();
gradientBoxMatrix.createGradientBox(200, 40, 0, 0, 0);
myShape.graphics.lineStyle(5, 0);
myShape.graphics.lineGradientStyle(GradientType.LINEAR, [0xFF0000, 0x00FF00, 0x0000FF], [1,
1, 1], [0, 128, 255], gradientBoxMatrix);
myShape.graphics.drawRect(0, 0, 200, 40);
this.addChild(myShape);
```

Weitere Informationen zur Matrix-Klasse finden Sie unter „Verwenden von Matrix-Objekten“ auf Seite 230.

Verwenden der Math-Klasse mit Zeichnungsmethoden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit einem Graphics-Objekt können Kreise und Quadrate, jedoch auch komplexere Formen gezeichnet werden, insbesondere bei Verwendung der Zeichnungsmethoden in Kombination mit den Eigenschaften und Methoden der Math-Klasse. Die Math-Klasse enthält Konstanten von allgemeinem mathematischen Interesse, z. B. `Math.PI` (rund 3.14159265...), eine Konstante für das Verhältnis zwischen dem Umfang und dem Durchmesser eines Kreises. Sie enthält auch Methoden für Trigonometriefunktionen, darunter u. a. `Math.sin()`, `Math.cos()` und `Math.tan()`. Beim Zeichnen von Formen mit diesen Methoden und Konstanten können dynamischere visuelle Effekte erzielt werden, vor allem bei Verwendung von Wiederholung oder Rekursion.

Bei vielen Methoden der Math-Klasse, mit Ausnahme kreisförmiger Maße, müssen Einheiten in Bogenmaß und nicht in Grad angegeben werden. Das Umrechnen zwischen diesen beiden Einheiten ist ein häufiger Verwendungszweck der Math-Klasse:

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```

Im folgenden Beispiel werden eine Sinuswelle und eine Kosinuswelle erstellt, um den Unterschied zwischen den Methoden `Math.sin()` und `Math.cos()` für einen angegebenen Wert zu verdeutlichen.

```
var sinWavePosition = 100;
var cosWavePosition = 200;
var sinWaveColor:uint = 0xFF0000;
var cosWaveColor:uint = 0x00FF00;
var waveMultiplier:Number = 10;
var waveStretcher:Number = 5;

var i:uint;
for(i = 1; i < stage.stageWidth; i++)
{
    var sinPosY:Number = Math.sin(i / waveStretcher) * waveMultiplier;
    var cosPosY:Number = Math.cos(i / waveStretcher) * waveMultiplier;

    graphics.beginFill(sinWaveColor);
    graphics.drawRect(i, sinWavePosition + sinPosY, 2, 2);
    graphics.beginFill(cosWaveColor);
    graphics.drawRect(i, cosWavePosition + cosPosY, 2, 2);
}
```

Animation mit der Zeichnungs-API

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Vorteil beim Erstellen von Inhalten mit der Zeichnungs-API besteht darin, dass die Positionierung des Inhalts nicht auf eine einmalige Positionierung beschränkt ist. Die gezeichneten Objekte können durch Festlegen und Bearbeiten der beim Zeichnen verwendeten Variablen geändert werden. Sie können Objekte animieren, indem Sie Variablen ändern und die Objekte dann neu zeichnen, entweder über mehrere Bilder über mithilfe eines Timers.

Mit dem folgenden Code wird beispielsweise die Anzeige mit jedem Bild geändert (durch Überwachen des `Event.ENTER_FRAME`-Ereignisses), die aktuelle Gradzählung inkrementiert sowie das Graphics-Objekt gelöscht und an der aktualisierten Position neu gezeichnet.

```
stage.frameRate = 31;

var currentDegrees:Number = 0;
var radius:Number = 40;
var satelliteRadius:Number = 6;

var container:Sprite = new Sprite();
container.x = stage.stageWidth / 2;
container.y = stage.stageHeight / 2;
addChild(container);
var satellite:Shape = new Shape();
container.addChild(satellite);

addEventListener(Event.ENTER_FRAME, doEveryFrame);

function doEveryFrame(event:Event):void
{
    currentDegrees += 4;
    var radians:Number = getRadians(currentDegrees);
    var posX:Number = Math.sin(radians) * radius;
    var posY:Number = Math.cos(radians) * radius;
    satellite.graphics.clear();
    satellite.graphics.beginFill(0);
    satellite.graphics.drawCircle(posX, posY, satelliteRadius);
}
function getRadians(degrees:Number):Number
{
return degrees * Math.PI / 180;
}
```

Um ein erheblich abweichendes Ergebnis zu erzielen, können Sie die ursprünglichen Anfangsvariablen `currentDegrees`, `radius` und `satelliteRadius` am Anfang des Codes ändern. Verringern Sie beispielsweise den Wert für die radius-Variable, und/oder erhöhen Sie den Wert für die totalSatellites-Variable. Hierbei handelt es sich nur um ein Beispiel dafür, wie mithilfe der Zeichnungs-API visuelle Darstellungen erstellt werden können, deren Komplexität die Einfachheit ihrer Erstellung verdecken.

Beispiel für die Zeichnungs-API: Algorithmic Visual Generator

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Beispielanwendung „Algorithmic Visual Generator“ werden auf der Bühne mehrere „Satelliten“ oder Kreise in einer kreisförmigen Umlaufbahn gezeichnet. Folgende Funktionen werden dabei erläutert:

- Zeichnen einer Grundform mit dynamischem Erscheinungsbild mithilfe der Zeichnungs-API
- Verknüpfen der Benutzerinteraktion mit den Eigenschaften einer Zeichnung
- Animation durch Löschen der Bühne und Neuzeichnen der Objekte für jedes Einzelbild

Im Beispiel im vorherigen Abschnitt wurde ein einzelner Satellit mithilfe des `Event.ENTER_FRAME`-Ereignisses animiert. Dieses Beispiel wird mit dieser Beispielanwendung erweitert, indem ein Bedienfeld mit mehreren Schiebereglern hinzugefügt wird, mit denen die visuelle Anzeige mehrerer Satelliten unmittelbar aktualisiert wird. In diesem Beispiel wird der Code als externe Klassen formalisiert und die Erstellung des Satelliten in eine Schleife eingebettet. Dabei wird jeweils ein Verweis auf die einzelnen Satelliten in einem `satellites`-Array gespeichert.

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Anwendungsdateien befinden sich im Ordner „Samples/AlgorithmicVisualGenerator“. Dieser Ordner enthält die folgenden Dateien:

Datei	Beschreibung
AlgorithmicVisualGenerator.fla	Die Hauptanwendungsdatei in Flash Professional (FLA).
com/example/programmingas3/algorithmic/AlgorithmicVisualGenerator.as	Die Klasse mit den Hauptfunktionen der Anwendung, einschließlich Zeichnen der Satelliten auf der Bühne und Reagieren auf Ereignisse über das Bedienfeld zum Aktualisieren der Variablen, die sich auf das Zeichnen der Satelliten auswirken.
com/example/programmingas3/algorithmic/ControlPanel.as	Eine Klasse, mit der die Benutzerinteraktion mit mehreren Schiebereglern verwaltet sowie bei auftretender Benutzerinteraktion Ereignisse ausgelöst werden.
com/example/programmingas3/algorithmic/Satellite.as	Eine Klasse, die das Anzeigeobjekt repräsentiert, das sich in einer Umlaufbahn um einen zentralen Punkt dreht und die Eigenschaften des aktuellen Zeichnungsstatus enthält.

Festlegen der Listener

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Anwendung werden zunächst drei Listener erstellt. Mit dem ersten Listener wird ein über das Bedienfeld ausgelöstes Ereignis überwacht, das angibt, dass die Neuerstellung der Satelliten erforderlich ist. Mit dem zweiten Listener werden Änderungen überwacht, die an der Größe der Bühne der SWF-Datei vorgenommen werden. Mit dem dritten Listener werden die einzelnen Bilder in der SWF-Datei überwacht und mithilfe der `doEveryFrame()`-Funktion neu gezeichnet.

Erstellen der Satelliten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Nach dem Einrichten der Listener wird die `build()`-Funktion aufgerufen. In dieser Funktion wird zunächst die `clear()`-Funktion aufgerufen, mit der das `satellites`-Array geleert wird und alle vorherigen Zeichnungen auf der Bühne gelöscht werden. Dies ist erforderlich, da die `build()`-Funktion jedes Mal neu aufgerufen werden kann, wenn über das Bedienfeld eine entsprechende Anweisung gesendet wird, z. B. nach dem Ändern der Farbeinstellungen. In diesem Fall müssen die Satelliten entfernt und neu erstellt werden.

In der Funktion werden dann die Satelliten erstellt. Dabei werden die ursprünglichen für die Erstellung erforderlichen Eigenschaften festgelegt, z. B. die `position`-Variable, die an einer zufällig gewählten Position in der Umlaufbahn startet, oder die `color`-Variable, die sich in diesem Beispiel nach dem Erstellen des Satelliten nicht mehr ändert.

Nach dem Erstellen der einzelnen Satelliten wird dem `satellites`-Array jeweils ein entsprechender Verweis hinzugefügt. Beim Aufruf der `doEveryFrame()`-Funktion werden alle Satelliten in diesem Array aktualisiert.

Aktualisieren der Satellitenposition

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `doEveryFrame()`-Funktion ist das Kernstück des Animationsvorgangs in der Anwendung. Sie wird bei jedem Bild mit einer Rate aufgerufen, die der Bildrate der SWF-Datei entspricht. Die Animation wird durch die geringfügige Änderung der Variablen in der Zeichnung umgesetzt.

Mit dieser Funktion werden zunächst alle vorherigen Zeichnungsobjekte gelöscht und die Hintergrundobjekte neu gezeichnet. Anschließend werden die einzelnen Satellitencontainer durchlaufen und die `position`-Eigenschaft jedes Satelliten inkrementiert sowie die Eigenschaften `radius` und `orbitRadius` aktualisiert, die durch die Benutzerinteraktion über das Bedienfeld möglicherweise geändert wurden. Schließlich wird die neue Position der Satelliten durch Aufruf der `draw()`-Methode der Satellite-Klasse aktualisiert.

Beachten Sie, dass der Zähler `i` nur bis zur `visibleSatellites`-Variablen inkrementiert wird. Dies ist darauf zurückzuführen, dass der Benutzer die Anzahl der angezeigten Satelliten über das Bedienfeld begrenzen kann. In diesem Fall dürfen die übrigen Satelliten in der Schleife nicht neu gezeichnet, sondern müssen ausgeblendet werden. Dies erfolgt in einer Schleife, die unmittelbar auf die Schleife folgt, mit der die Elemente gezeichnet werden.

Nach dem Abschluss der doEveryFrame()-Funktion wird die jeweilige Position der festgelegten `visibleSatellites`-Objekte auf dem Bildschirm aktualisiert.

Reagieren auf Benutzerinteraktionen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Benutzerinteraktionen erfolgen über das Bedienfeld, das über die ControlPanel-Klasse verwaltet wird. Mit dieser Klasse wird ein Listener mit den entsprechenden minimalen, maximalen und Standardwerten für die einzelnen Schieberegler festgelegt. Wenn der Benutzer diese Schieberegler verschiebt, wird die `changeSetting()`-Funktion aufgerufen. Mit dieser Funktion werden die Eigenschaften des Bedienfelds aktualisiert. Wenn die Anzeige aufgrund der Änderungen neu erstellt werden muss, wird ein entsprechendes Ereignis ausgelöst, das dann in der Hauptanwendungsdatei verarbeitet wird. Mit jeder Änderung an den Einstellungen für das Bedienfeld wird über die `doEveryFrame()`-Funktion jeder Satellit mit den aktualisierten Variablen neu gezeichnet.

Weitere Anpassungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei diesem Beispiel handelt es sich lediglich um einen Grundbauplan für das Erzeugen von visuellen Effekten mit der Zeichnungs-API. Mit verhältnismäßig wenigen Codezeilen wird eine interaktive Anwendung erstellt, die recht komplex wirkt. Dennoch kann dieses Beispiel durch geringfügige Änderungen noch wesentlich erweitert werden. Es folgen einige Vorschläge:

- Beispielsweise kann in der `doEveryFrame()`-Funktion der Farbwert der Satelliten inkrementiert werden.
- In der `doEveryFrame()`-Funktion kann der Radius der Satelliten über einen bestimmten Zeitraum verkleinert oder vergrößert werden.
- Der Satellitenradius muss nicht kreisförmig sein. Mithilfe der Math-Klasse kann die Bewegung beispielsweise entlang einer Sinuskurve erfolgen.

- Es kann die Kollision von Satelliten mit anderen Satelliten abgefragt werden.

Die Zeichnungs-API kann als Alternative zum Erstellen visueller Effekte in der Flash-Authoring-Umgebung verwendet werden, um Grundformen zur Laufzeit zu erstellen. Darüber hinaus können visuelle Effekte in einer Vielfalt und einem Umfang erzielt werden, die manuell nicht möglich sind. Mithilfe der Zeichnungs-API und elementaren Mathematikkenntnissen können Sie in ActionScript unzählige unerwartete Formen kreieren.

Erweiterte Einsatzmöglichkeiten der Zeichnungs-API

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Flash Player 10, Adobe AIR 1.5 und höhere Versionen der Flash-Laufzeitumgebungen unterstützen einen erweiterten Satz von Zeichnungsfunktionen. Die Verbesserungen der Zeichnungs-API für diese Laufzeitumgebungen erweitern die Zeichnungsmethoden früherer Versionen; mit den neuen Funktionen können Sie Datensätze einrichten, um Formen zu generieren, Formen zur Laufzeit ändern und dreidimensionale Effekte erstellen. Die Erweiterungen der Zeichnungs-API konsolidieren vorhandene Methoden in alternative Befehle. Diese Befehle nutzen Vektor-Arrays und Aufzählungsklassen, um Datensätze für Zeichnungsmethoden bereitzustellen. Mithilfe von Vektor-Arrays können komplexe Formen schneller dargestellt werden. Entwickler können die Array-Werte programmgesteuert ändern, um Formen zur Laufzeit dynamisch darzustellen.

Die in Flash Player 10 eingeführten Zeichnungsfunktionen werden in den folgenden Abschnitten beschrieben: „Zeichenpfade“ auf Seite 251, „Definieren von Windungsregeln“ auf Seite 252, „Verwenden von Graphics-Datenklassen“ auf Seite 254 und „Verwenden von „drawTriangles()““ auf Seite 257.

Im Folgenden sind Aufgaben aufgeführt, die Sie mit der erweiterten Zeichnungs-API in ActionScript ausführen können:

- Verwenden von Vector-Objekten zum Speichern von Daten für Zeichenmethoden
- Definieren von Pfaden für das programmgesteuerte Zeichnen von Formen in einem Vorgang
- Definieren von Windungsregeln, um zu bestimmen, wie überlappende Formen gefüllt werden
- Lesen der Vektorgrafikinhalte eines Anzeigeobjekts, um z. B. die Grafikdaten zu serialisieren und zu speichern, Spritesheets zur Laufzeit zu erstellen oder eine Kopie der Vektorgrafikinhalte zu zeichnen
- Verwenden von Dreiecken und Zeichenmethoden für dreidimensionale Effekte

Wichtige Konzepte und Begriffe

Im Folgenden sind wichtige Begriffe aufgeführt, die in diesem Abschnitt verwendet werden:

- Vektor: Ein Array von Werten, die denselben Datentyp aufweisen. Ein Vector-Objekt kann ein Array von Werten speichern, die Zeichenmethoden verwenden, um Linien und Formen mit einem einzigen Befehl zu erstellen. Weitere Informationen zu Vector-Objekten finden Sie unter „Indizierte Arrays“ auf Seite 27.
- Pfad: Ein Pfad besteht aus einem oder mehreren geraden oder gekrümmten Segmenten. Anfang und Ende jedes Segments sind durch Koordinaten gekennzeichnet, die man sich als Heftzwecken vorstellen kann, die eine Schnur fixieren. Ein Pfad ist entweder geschlossen (z. B. ein Kreis) oder geöffnet (mit eindeutigen Endpunkten; z. B. eine Wellenlinie).
- Windung: Die Richtung eines Pfades, wie sie vom Renderer interpretiert wird; entweder positiv (im Uhrzeigersinn) oder negativ (entgegen dem Uhrzeigersinn).

- GraphicsStroke: Eine Klasse zum Einstellen des Linienstils. Der Begriff „stroke“ (Strich) ist kein Teil der Erweiterungen der Zeichnungs-API, die Verwendung einer Klasse zum Bestimmen eines Linienstils mit einer eigenen Fülleigenschaft ist jedoch Teil der neuen Zeichnungs-API. Mit der GraphicsStroke-Klasse können Sie den Stil einer Linie dynamisch anpassen.
- Fill-Objekt: Objekte, die mit Anzeigeklassen wie flash.display.GraphicsBitmapFill und flash.display.GraphicsGradientFill erstellt werden und an den Zeichenbefehl `Graphics.drawGraphicsData()` übergeben werden. Fill-Objekte und die erweiterten Zeichenbefehle führen einen objektorientierten Programmierungsansatz für das Replizieren von `Graphics.beginBitmapFill()` und `Graphics.beginGradientFill()` ein.

Zeichenpfade

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Im Abschnitt zum Zeichnen von Linien und Kurven (siehe „Zeichnen von Linien und Kurven“ auf Seite 238) wurden die Befehle für das Zeichnen einer einzelnen Linie (`Graphics.lineTo()`) oder Kurve (`Graphics.curveTo()`) und für das Verschieben der Linie an einen anderen Punkt (`Graphics.moveTo()`) zum Erstellen von Formen vorgestellt. Die Methoden `Graphics.drawPath()` und `Graphics.drawTriangles()` akzeptieren eine Reihe von Objekten, die dieselben Zeichenbefehle als Parameter darstellen. Mit diesen Methoden können Sie eine Reihe von `Graphics.lineTo()`-, `Graphics.curveTo()`- oder `Graphics.moveTo()`-Befehlen für die Ausführung in der Flash-Laufzeitumgebung in einer einzigen Anweisung bereitstellen.

Die GraphicsPathCommand-Aufzählungsklasse definiert mehrere Konstanten, die Zeichenbefehlen entsprechen. Sie übergeben mehrere dieser Konstanten (in einer Vector-Instanz) als Parameter für die `Graphics.drawPath()`-Methode. Mit einem einzelnen Befehl können Sie dann eine ganze Form oder mehrere Formen darstellen. Sie können die an diese Methoden übergebenen Werte auch ändern, um eine vorhandene Form zu modifizieren.

Zusätzlich zu dem Vektor von Zeichenbefehlen erfordert die `drawPath()`-Methode eine Reihe von Koordinaten, die den Koordinaten für jeden Zeichenbefehl entsprechen. Erstellen Sie eine Vector-Instanz mit Koordinaten (Number-Instanzen) und übergeben Sie sie an die `drawPath()`-Methode als zweites (`data`) Argument.

Hinweis: *Die Werte im Vektor sind keine Point-Objekte; der Vektor besteht aus einer Reihe von Zahlen, wobei jeweils zwei Zahlen ein x/y-Koordinatenpaar darstellen.*

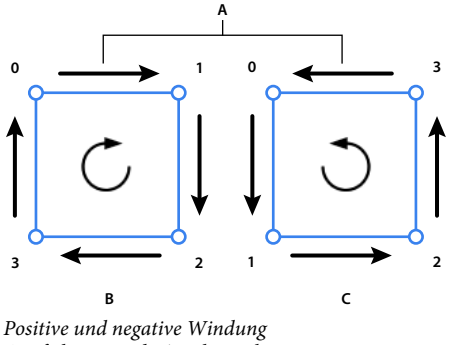
Die `Graphics.drawPath()`-Methode ordnet jeden Befehl den jeweiligen Punktwerten zu (einer Gruppe von zwei oder vier Zahlen), um im Graphics-Objekt einen Pfad zu erstellen:

```
package
{
    import flash.display.*;

    public class DrawPathExample extends Sprite
    {
        public function DrawPathExample(){

            var squareCommands:Vector.<int> = new Vector.<int>(5, true);
            squareCommands[0] = GraphicsPathCommand.MOVE_TO;
            squareCommands[1] = GraphicsPathCommand.LINE_TO;
            squareCommands[2] = GraphicsPathCommand.LINE_TO;
            squareCommands[3] = GraphicsPathCommand.LINE_TO;
            squareCommands[4] = GraphicsPathCommand.LINE_TO;

            var squareCoord:Vector.<Number> = new Vector.<Number>(10, true);
            squareCoord[0] = 20; //x
            squareCoord[1] = 10; //y
            squareCoord[2] = 50;
            squareCoord[3] = 10;
            squareCoord[4] = 50;
            squareCoord[5] = 40;
            squareCoord[6] = 20;
            squareCoord[7] = 40;
            squareCoord[8] = 20;
            squareCoord[9] = 10;

            graphics.beginFill(0x442266);//set the color
            graphics.drawPath(squareCommands, squareCoord);
        }
    }
}
```

Definieren von Windungsregeln

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die erweiterte Zeichnungs-API führt auch das Konzept der „Pfadwindung“ ein; dies ist die Richtung eines Pfades. Die Windung eines Pfades ist entweder positiv (im Uhrzeigersinn) oder negativ (entgegen dem Uhrzeigersinn). Die Windung wird bestimmt durch die Reihenfolge, in der der Renderer die vom Vektor des data-Parameters bereitgestellten Koordinaten interpretiert.



Positive und negative Windung
A. *Pfeile zeigen die Zeichenrichtung an* ***B.*** *Positiv gewunden (im Uhrzeigersinn)* ***C.*** *Negativ gewunden (gegen den Uhrzeigersinn)*

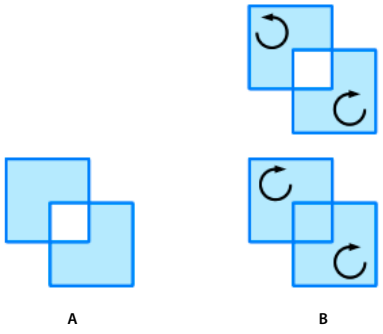
Beachten Sie, dass die `Graphics.drawPath()`-Methode über einen optionalen dritten Parameter, „winding“, verfügt:

```
drawPath(commands:Vector.<int>, data:Vector.<Number>, winding:String = "evenOdd"):void
```

In diesem Zusammenhang ist der dritte Parameter ein String oder eine Konstante, die die Windungs- oder Füllregel für sich überschneidende Teile angibt. (Die Konstantenwerte sind in der GraphicsPathWinding-Klasse als `GraphicsPathWinding.EVEN_ODD` oder `GraphicsPathWinding.NON_ZERO` definiert.) Die Windungsregel ist wichtig, wenn sich Pfade überschneiden.

Die Gerade-Ungerade-Regel ist die Standardwindungsregel. Sie wird von der herkömmlichen Zeichnungs-APIs verwendet. Die Gerade-Ungerade-Regel ist auch die Standardregel für die `Graphics.drawPath()`-Methode. Mit der Gerade-Ungerade-Windungsregel wechseln sich überschneidende Pfade zwischen offenen und geschlossenen Füllungen ab. Wenn sich zwei Quadrate mit derselben Füllung überschneiden, wird der Bereich der Überschneidung gefüllt. Benachbarte Bereiche sind niemals beide gefüllt oder beide nicht gefüllt.

Die Nicht-Null-Windungsregel berücksichtigt dagegen die Windung (Zeichenrichtung), um festzustellen, welche durch sich überschneidende Pfade definierten Bereiche gefüllt werden. Wenn Pfade mit unterschiedlichen Windungen sich überschneiden, bleibt der entstehende Bereich ungefüllt wie bei der Gerade-Ungerade-Regel. Bei Pfaden mit gleicher Windungsrichtung wird der entstehende Bereich gefüllt:

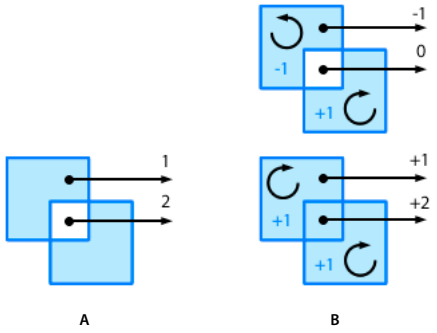


Windungsregeln für sich überlappende Bereiche
A. *Windungsregel „Gerade-Ungerade“* ***B.*** *Windungsregel „Nicht null“*

Namen der Windungsregeln

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Namen verweisen auf eine spezifischere Regel, die die Verwaltung dieser Füllungen definiert. Positiv gewundenen Pfaden wird der Wert +1 zugewiesen; negativ gewundenen Pfaden wird der Wert -1 zugewiesen. Zeichnen Sie eine Linie, die an einem Punkt in einem eingeschlossenen Bereich einer Form beginnt und sich von dort aus endlos nach außen fortsetzt. Mithilfe der Häufigkeit, mit der diese Linie einen Pfad kreuzt, und der kombinierten Werte dieser Pfade wird die Füllung bestimmt. Bei der Windungsregel „Gerade-Ungerade“ wird die Häufigkeit, mit der die Linie einen Pfad kreuzt, angewendet. Bei einer ungeraden Anzahl wird der Bereich gefüllt, während der Bereich bei einer geraden Anzahl nicht gefüllt wird. Bei der Windungsregel „Nicht null“ werden die den Pfaden zugewiesenen Werte angewendet. Ergeben die kombinierten Werte des Pfads nicht 0, wird der Bereich gefüllt. Ergeben die kombinierten Werte 0, wird der Bereich nicht gefüllt.



Anzahlen und Füllungen bei Windungsregeln
A. *Windungsregel „Gerade-Ungerade“* ***B.*** *Windungsregel „Nicht null“*

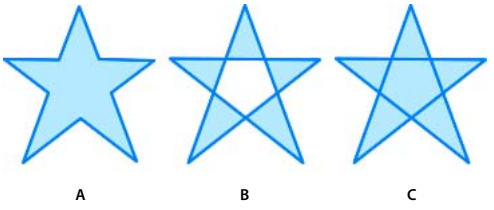
Verwenden von Windungsregeln

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Diese Füllungsregeln sind kompliziert, aber in manchen Situationen erforderlich. Nehmen wir beispielsweise an, Sie zeichnen eine Sternform. Bei der standardmäßigen Regel „Gerade-Ungerade“ würde die Form zehn verschiedene Linien erfordern. Mithilfe der Windungsregel „Nicht null“ werden diese zehn Linien auf fünf reduziert. Unten sehen Sie den ActionScript-Code für einen Stern mit fünf Linien und einer Windungsregel „Nicht null“:

```
graphics.beginFill(0x60A0FF);
graphics.drawPath( Vector.<int>([1,2,2,2,2]),  Vector.<Number>([66,10, 23,127, 122,50, 10,49,
109,127]),   GraphicsPathWinding.NON_ZERO);
```

Und dies ist die Sternform:



Eine Sternform mit einer anderen Windungsregel
A. *Gerade-Ungerade 10 Linien* ***B.*** *Gerade-Ungerade 5 Linien* ***C.*** *Nicht null 5 Linien*

Und da Bilder animiert werden oder als Texturen auf dreidimensionalen Objekten angewendet werden und überlappen, nehmen die Windungsregeln an Bedeutung zu.

Verwenden von Graphics-Datenklassen

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die optimierte Zeichnungs-API enthält eine Reihe von Klassen im flash.display-Paket, die die IGraphicsData-Schnittstelle implementieren. Diese Klassen agieren als Wertobjekte (Datencontainer), die die Zeichnungsmethoden der Zeichnungs-API darstellen.

Die folgenden Klassen implementieren die IGraphicsData-Schnittstelle:

- GraphicsBitmapFill
- GraphicsEndFill

- GraphicsGradientFill
- GraphicsPath
- GraphicsShaderFill
- GraphicsSolidFill
- GraphicsStroke
- GraphicsTrianglePath

Mit diesen Klassen können Sie vollständige Zeichnungen in einem Vector-Objekt des Typs IGraphicsData (Vector.<IGraphicsData>) speichern. Sie können die Grafikdaten anschließend als Datenquelle für andere Formen oder zum Speichern von Zeicheninformationen zur späteren Verwendung nutzen.

Beachten Sie, dass es für jeden Füllstil mehrere Füllklassen gibt, aber nur eine Strichklasse. ActionScript verfügt nur über eine IGraphicsData-Strichklassen, dass die Strichklasse die Füllklassen zur Definition des Stils verwendet. Somit ist jeder Strich durch eine Kombination aus Strichklasse und Füllklasse definiert. Die APIs für diese Graphics-Datenklassen spiegeln die Methoden, die sie repräsentieren, in der flash.display.Graphics-Klasse:

Graphics-Methode	**Entsprechende Klasse**
beginBitmapFill()	GraphicsBitmapFill
beginFill()	GraphicsSolidFill
beginGradientFill()	GraphicsGradientFill
beginShaderFill()	GraphicsShaderFill
lineBitmapStyle()	GraphicsStroke + GraphicsBitmapFill
lineGradientStyle()	GraphicsStroke + GraphicsGradientFill
lineShaderStyle()	GraphicsStroke + GraphicsShaderFill
lineStyle()	GraphicsStroke + GraphicsSolidFill
moveTo() lineTo() curveTo() drawPath()	GraphicsPath
drawTriangles()	GraphicsTrianglePath

Zusätzlich verfügt die GraphicsPath-Klasse über eigene Methoden `GraphicsPath.moveTo()`, `GraphicsPath.lineTo()`, `GraphicsPath.curveTo()`, `GraphicsPath.wideLineTo()` und `GraphicsPath.wideMoveTo()`, um diese Befehle einfach für eine GraphicsPath-Instanz zu definieren. Diese Dienstprogrammmethoden vereinfachen das direkte Definieren oder Aktualisieren der Befehle und Datenwerte.

Zeichnen mit Vektorgrafikdaten

Nachdem Sie über eine Reihe von IGraphicsData-Instanzen verfügen, verwenden Sie `drawGraphicsData()`-Methoden der Graphics-Klasse, um die Grafiken darzustellen. Die `drawGraphicsData()`-Methode führt nacheinander mehrere Zeichnungsanweisungen eines Vektors von IGraphicsData-Instanzen aus:

```
// stroke object
var stroke:GraphicsStroke = new GraphicsStroke(3);
stroke.joints = JointStyle.MITER;
stroke.fill = new GraphicsSolidFill(0x102020);// solid stroke

// fill object
var fill:GraphicsGradientFill = new GraphicsGradientFill();
fill.colors = [0x0000FF, 0xEEFFEE];
fill.matrix = new Matrix();
fill.matrix.createGradientBox(70, 70, Math.PI/2);
// path object
var path:GraphicsPath = new GraphicsPath(new Vector.<int>(), new Vector.<Number>());
path.commands.push(GraphicsPathCommand.MOVE_TO, GraphicsPathCommand.LINE_TO,
GraphicsPathCommand.LINE_TO);
path.data.push(125,0, 50,100, 175,0);

// combine objects for complete drawing
var drawing:Vector.<IGraphicsData> = new Vector.<IGraphicsData>();
drawing.push(stroke, fill, path);

// draw the drawing
graphics.drawGraphicsData(drawing);
```

Indem Sie einen Wert im Pfad, der von der Zeichnung verwendet wird, ändern, kann die Form mehrere Male neu gezeichnet werden, um ein komplexeres Bild zu erstellen:

```
// draw the drawing multiple times
// change one value to modify each variation
graphics.drawGraphicsData(drawing);
path.data[2] += 200;
graphics.drawGraphicsData(drawing);
path.data[2] -= 150;
graphics.drawGraphicsData(drawing);
path.data[2] += 100;
graphics.drawGraphicsData(drawing);
path.data[2] -= 50;graphicsS.drawGraphicsData(drawing);
```

IGraphicsData-Objekte können zwar Füll- und Strichstile definieren, diese Stile sind jedoch nicht unbedingt erforderlich. Anders ausgedrückt lassen sich Methoden der Graphics-Klasse verwenden, um Stile festzulegen, während IGraphicsData-Objekte verwendet werden, um eine gespeicherte Sammlung von Pfaden zu zeichnen oder umgekehrt.

Hinweis: *Verwenden Sie die `Graphics.clear()`-Methode, um eine vorherige Zeichnung zu löschen, bevor eine neue begonnen wird, es sei denn, Sie fügen der Originalzeichnung etwas hinzu wie im Beispiel oben. Wenn Sie einen Teil eines Pfades oder einer Sammlung von IGraphicsData-Objekten ändern, zeichnen Sie die gesamte Zeichnung neu, um die Änderungen zu sehen.*

Wenn Sie Graphics-Datenklassen verwenden, wird die Füllung dargestellt, wenn drei oder mehr Punkte gezeichnet werden, da die Form an diesem Punkt inherent geschlossen ist. Auch wenn die Füllung geschlossen wird, wird es der Strich nicht. Dieses Verhalten unterscheidet sich von der Verwendung der Befehle `Graphics.lineTo()` oder `Graphics.moveTo()`.

Lesen von Vektorgrafikdaten

Flash Player 11.6 und höher, Adobe AIR 3.6 und höher

Zusätzlich zum Zeichnen von Vektorinhalten in ein Anzeigeobjekt können Sie in Flash Player 11.6 und Adobe AIR 3.6 (und höher) mithilfe der `readGraphicsData()`-Methode der Graphics-Klasse eine Datendarstellung der Vektorgrafikinhalte eines Anzeigeobjekts abrufen. Sie können damit einen Schnappschuss einer Grafik erstellen, um etwa zur Laufzeit ein Spritesheet zu speichern, zu kopieren oder zu erstellen.

Durch das Abrufen der `readGraphicsData()`-Methode wird eine Vector-Instanz mit IGraphicsData-Objekten zurückgegeben. Es handelt sich dabei um dieselben Objekte, mit denen mithilfe der `drawGraphicsData()`-Methode Vektorgrafiken gezeichnet wurden.

Es bestehen jedoch mehrere Einschränkungen für das Lesen von Vektorgrafiken mit der `readGraphicsData()`-Methode. Weitere Informationen finden Sie im readGraphicsData()-Eintrag im ActionScript-Referenzhandbuch.

Verwenden von „drawTriangles()“

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Eine andere erweiterte Methode, die ab Flash Player 10 und Adobe AIR 1.5 zur Verfügung steht, ist `Graphics.drawTriangles()`. Sie ähnelt der `Graphics.drawPath()`-Methode. Die `Graphics.drawTriangles()`-Methode verwendet ebenfalls ein Vector.<Number>-Objekt, um Punktpositionen beim Zeichnen von Pfaden anzugeben.

Der eigentliche Zweck der `Graphics.drawTriangles()`-Methode besteht allerdings darin, die Verwendung dreidimensionaler Effekte mit ActionScript zu erleichtern. Informationen zur Verwendung von `Graphics.drawTriangles()` für dreidimensionale Effekte finden Sie unter „Verwenden von Dreiecken für 3D-Effekte“ auf Seite 388.

Kapitel 13: Verwenden von Bitmaps

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Neben den Funktionen für das Zeichnen von Vektorgrafik enthält ActionScript 3.0 Funktionen zum Erstellen von Bitmapbildern oder zum Bearbeiten der Pixeldaten externer Bitmapbilder, die in SWF-Dateien geladen werden. Aufgrund der Möglichkeit zum Abrufen und Ändern einzelner Pixelwerte können Sie benutzerdefinierte filterähnliche Bildeffekte sowie mithilfe der integrierten Störungsfunktionen Texturen und zufällige Störungen erstellen.

- Renaun Erickson: Rendering game assets in ActionScript using blitting techniques
- Bitmap programming: Kapitel 26 des Handbuchs „Essential ActionScript 3“ von Colin Moock (O'Reilly Media, 2007)
- Mike Jones: Working with Sprites in Pushbutton Engine
- Flash & Math: Pixel Particles Made Simple
- Flixel

Grundlagen zum Arbeiten mit Bitmaps

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei der Bearbeitung digitaler Bilder begegnen Ihnen wahrscheinlich zwei Arten von Grafiken: Bitmaps und Vektoren. Bitmapgrafiken, die auch als Rastergrafiken bezeichnet werden, setzen sich aus winzigen Quadraten (Pixeln) zusammen, die in einer rechteckigen Rasterstruktur angeordnet sind. Vektorgrafiken bestehen aus mathematisch erstellten geometrischen Formen, z. B. Linien, Kurven und Polygonen.

Bitmapbilder sind durch die Breite und Höhe des Bilds in Pixel sowie durch die Bit pro Pixel definiert, deren Anzahl Auswirkungen auf die Anzahl der möglichen Farben pro Pixel hat. Bei einer Bitmapgrafik mit dem RGB-Farbmodell setzen sich die Pixel aus je drei Byte zusammen: Rot, Grün und Blau. Jedes Byte enthält einen Wert im Bereich zwischen 0 und 255. Wenn sie in einem Pixel kombiniert werden, entsteht ähnlich wie beim Mischen von Farben ein bestimmter Farbton. Ein Pixel mit den Bytewerten Rot=255, Grün=102 und Blau=0 ergibt beispielsweise einen kräftigen Orangeton.

Die Qualität eines Bitmapbilds setzt sich aus der Bildauflösung und dem entsprechenden Bitwert für die Farbtiefe zusammen. *Auflösung* bezieht sich auf die Anzahl der Pixel in einem Bild. Je höher die Anzahl der Pixel, desto höher die Auflösung und desto nuancierter die Darstellung des Bilds. *Farbtiefe* bezieht sich auf die je Pixel kodierbare Informationsmenge. In einem Bild mit einer Farbtiefe von 16 Bit pro Pixel kann beispielsweise nicht die gleiche Anzahl von Farben dargestellt werden wie in einem Bild mit einer Farbtiefe von 48 Bit. Das Bild mit einer Farbtiefe von 48 Bit weist daher feinere Schattierungsabstufungen auf als das Bild mit 16 Bit.

Da Bitmapgrafiken von der Auflösung abhängen, lassen sie sich nicht sehr gut skalieren. Dies macht sich am meisten beim Vergrößern von Bitmapbildern bemerkbar. Bei vergrößerten Bitmapgrafiken verschlechtern sich in der Regel der Detailgrad und die Qualität.

Dateiformate von Bitmapgrafiken

Bitmapbilder können in mehreren häufig verwendeten Dateiformaten vorliegen. Bei diesen Formaten werden je nach Verwendungszweck eines Bilds unterschiedliche Komprimierungsalgorithmen zur Reduzierung der Dateigröße sowie zur Optimierung der Bildqualität verwendet. Adobe-Laufzeitumgebungen unterstützen die Bitmapformate BMP, GIF, JPG, PNG und TIFF.

BMP

Das BMP-Format (Bit Mapped) ist ein Standardbildformat, das vom Microsoft Windows-Betriebssystem verwendet wird. Hierbei wird keinerlei Komprimierungsalgorithmus verwendet und es führt daher im Allgemeinen zu sehr großen Dateien.

GIF

Das GIF-Format (Graphics Interchange Format) wurde ursprünglich 1987 von CompuServe zum Übertragen von Bildern mit 256 Farben (8-Bit-Farbtiefe) entwickelt. Das Format ermöglicht kleine Dateigrößen und eignet sich optimal für webbasierte Bilder. Aufgrund der begrenzten Farbpalette dieses Formats eignen sich GIF-Bilder in der Regel nicht für Fotos, die normalerweise einen hohen Grad an Schattierungen und Farbverläufen erfordern. GIF-Bilder ermöglichen die Angabe der Transparenz für ein einzelnes Bit, sodass Farben als durchsichtig (oder transparent) definiert werden können. Dadurch scheint die Hintergrundfarbe einer Webseite an den Bereichen eines Bilds durch, für die Transparenz festgelegt wurde.

JPEG

Dieses Bildformat wurde von der Joint Photographic Experts Group (JPEG) entwickelt. Beim JPEG-Bildformat (häufig auch als JPG bezeichnet) werden mithilfe eines verlustbehafteten Komprimierungsalgorithmus eine 24-Bit-Farbtiefe und eine kleine Dateigröße ermöglicht. Verlustbehaftete Komprimierung bedeutet, dass jeder Speichervorgang eines Bilds zwar zu einer kleineren Dateigröße führt, jedoch auch einen Qualitäts- und Informationsverlust nach sich zieht. Das JPEG-Format eignet sich hervorragend für Fotos, da Millionen von Farben dargestellt werden können. Durch die Möglichkeit, den Komprimierungsgrad eines Bilds zu steuern, können Sie die Bildqualität und die Dateigröße beeinflussen.

PNG

Das PNG-Format (Portable Network Graphics) wurde als Open Source-Alternative zum patentierten GIF-Dateiformat entwickelt. PNG-Dateien unterstützen eine Farbtiefe von bis zu 64 Bit. Damit sind bis zu 16 Millionen Farben darstellbar. Da es sich bei PNG um ein relativ neues Format handelt, werden PNG-Dateien in einigen älteren Browsern nicht unterstützt. Im Gegensatz zum JPG-Format handelt es sich bei PNG um ein Bildformat mit verlustfreier Komprimierung, sodass beim Speichern eines Bilds keine Bildinformationen verloren gehen. PNG-Dateien unterstützen zudem Alphatransparenz, mit der bis zu 256 Transparenzabstufungen möglich sind.

TIFF

Vor der Einführung von PNG war TIFF (Tagged Image File Format) das beliebteste plattformübergreifende Format. Der Nachteil beim TIFF-Format besteht in seinen vielen Varianten, denn es gibt keinen Reader, der alle Versionen verarbeiten kann. Außerdem wird das Format gegenwärtig von keinem Webbrowser unterstützt. TIFF kann mit verlustreicher und verlustfreier Komprimierung eingesetzt werden und kann gerätespezifische Farbräume (wie etwa CMYK) verarbeiten.

Transparente Bitmaps und undurchsichtige Bitmaps

Bei Bitmapbildern im GIF- oder PNG-Format kann jedem Pixel ein zusätzlicher Bytewert (Alphakanal) hinzugefügt werden. Dieses zusätzliche Pixelbyte stellt den Transparenzwert des entsprechenden Pixels dar.

GIF-Bilder ermöglichen die Transparenz für ein einzelnes Bit. Das bedeutet, Sie können für eine einzelne Farbe aus einer Farbpalette mit 256 Farben festlegen, dass diese transparent dargestellt wird. PNG-Bilder können dagegen bis zu 256 Transparenzabstufungen aufweisen. Diese Funktion ist vor allem dann von Vorteil, wenn in Bildern oder Texten der Hintergrund durchscheinen soll.

In ActionScript 3.0 wird dieses zusätzliche Pixelbyte für die Transparenz in der BitmapData-Klasse abgebildet. Analog zum PNG-Transparenzmodell können mit ActionScript bis zu 256 Transparenzgrade angegeben werden.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe aufgeführt, die Ihnen im Zusammenhang mit Bitmap-Grafiken häufig begegnen:

Alpha Der Transparenzgrad (oder genauer die Opazität) in einer Farbe oder einem Bild. Der Alphabetrag wird häufig als Wert des *Alphakanals* bezeichnet.

ARGB-Farbe Ein Farbschema, bei dem die Farbe jedes Pixels aus einer Mischung der Rot-, Grün- und Blaufarbwerte besteht und die entsprechende Transparenz mit einem Alphawert angegeben wird.

Farbkanal In der Regel werden Farben als Mischung einiger Grundfarben dargestellt, (bei Computergrafiken) im Allgemeinen Rot, Grün und Blau. Jede Grundfarbe wird als Farbkanal betrachtet. Durch den Farbanteil in jedem Farbkanal wird die endgültige Farbe festgelegt.

Farbtiefe Wird auch als *Bittiefe* bezeichnet und bezieht sich auf den Speicherplatz auf dem Computer, den jedes Pixel belegt. Sie bestimmt die Anzahl der möglichen Farben, die in einem Bild dargestellt werden können.

Pixel Die kleinste Dateneinheit in einem Bitmapbild. Es handelt sich im Wesentlichen um einen Farbpunkt.

Auflösung Die Abmessungen eines Bilds in Pixel, die den Detailgrad im Bild bestimmen. Die Auflösung wird häufig durch die Breite und Höhe als Anzahl an Pixeln angegeben.

RGB-Farbe Ein Farbschema, bei dem die Farbe jedes Pixels als Mischung der Farben Rot, Grün und Blau dargestellt wird.

Bitmap-Klasse und BitmapData-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die wichtigsten ActionScript 3.0-Klassen für die Arbeit mit Bitmaps sind die Bitmap-Klasse, die der Anzeige von Bitmaps auf der Bühne dient, und die BitmapData-Klasse, die für den Zugriff auf und die Bearbeitung von Bildrohdaten einer Bitmap verwendet wird.

Verwandte Hilfethemen

flash.display.Bitmap

flash.display.BitmapData

Bitmap-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Als Unterklasse der DisplayObject-Klasse handelt es sich bei der Bitmap-Klasse um die Hauptklasse von ActionScript 3.0 für die Anzeige von Bitmapbildern. Diese Bilder können über die flash.display.Loader-Klasse geladen oder mit dem `Bitmap()`-Konstruktor dynamisch erstellt worden sein. Beim Laden einer Bitmapgrafik aus einer externen Quelle können als Bitmap-Objekte nur Bilder im Format GIF, JPEG oder PNG verwendet werden. Nach der Instanziierung kann die Bitmap-Instanz als Wrapper eines BitmapData-Objekts eingesetzt werden, das auf der Bühne dargestellt werden soll. Da es sich bei Bitmap-Instanzen um Anzeigeobjekte handelt, können Bitmap-Instanzen auch mit allen Eigenschaften und Funktionen von Anzeigeobjekten bearbeitet werden. Weitere Informationen zur Verwendung von Anzeigeobjekten finden Sie unter „Programmieren von Anzeigeobjekten“ auf Seite 161.

Pixelausrichtung und -glättung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Neben den Funktionen, die in allen Anzeigeobjekten verfügbar sind, werden mit der Bitmap-Klasse einige zusätzliche für Bitmapbilder spezifische Funktionen bereitgestellt.

Die `pixelSnapping`-Eigenschaft der Bitmap-Klasse bestimmt, ob ein Bitmap-Objekt an seinen nächsten Pixeln ausgerichtet wird oder nicht. Für diese Eigenschaft kann eine der drei in der PixelSnapping-Klasse definierten Konstanten angegeben werden: `ALWAYS`, `AUTO` und `NEVER`.

Die Syntax für die Pixelausrichtung lautet wie folgt:

```
myBitmap.pixelSnapping = PixelSnapping.ALWAYS;
```

Nach dem Skalieren sind Bitmapbilder häufig verschwommen und verzerrt. Verzerrungen können mithilfe der `smoothing`-Eigenschaft der BitmapData-Klasse reduziert werden. Wenn diese boolesche Eigenschaft auf `true` gesetzt ist, werden die Pixel im Bild beim Skalieren geglättet (Anti-Aliasing). Dadurch wird das Bild klarer und natürlicher dargestellt.

BitmapData-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die im flash.display-Paket enthaltene BitmapData-Klasse ist mit einem fotografischen Schnappschuss der Pixel vergleichbar, die in einem geladenen oder dynamisch erstellten Bitmapbild vorhanden sind. Der Schnappschuss wird als Array von Pixeldaten im Objekt abgebildet. Die BitmapData-Klasse enthält zudem mehrere integrierte Methoden, die hilfreich zum Erstellen und Bearbeiten von Pixeldaten sind.

Verwenden Sie zum Instanziieren eines BitmapData-Objekts den folgenden Code:

```
var myBitmap:BitmapData = new BitmapData(width:Number, height:Number, transparent:Boolean,
fillColor:uinit);
```

Die Parameter `width` und `height` geben die Größe der Bitmap an. Beginnend mit AIR 3 und Flash Player 11 wurden die Größenbegrenzungen für BitmapData-Objekte entfernt. Die maximale Bitmapgröße wird durch das Betriebssystem festgelegt.

In AIR 1.5 und Flash Player 10 beträgt die maximale Höhe oder Breite eines BitmapData-Objekts 8.191 Pixel, die gesamte Pixelzahl darf 16.777.215 nicht übersteigen. (Wenn ein BitmapData-Objekt also 8.191 Pixel breit ist, darf es nur 2.048 Pixel hoch sein.) In Flash Player 9 und niedriger und AIR 1.1 und niedriger liegt die Grenze bei je 2.880 Pixel Höhe und Breite.

Mit dem `transparent`-Parameter wird angegeben, ob die Bitmapdaten einen Alphakanal enthalten (`true`) oder nicht (`false`). Der `fillColor`-Parameter ist ein 32-Bit-Farbwert, mit dem die Hintergrundfarbe sowie der Transparenzwert (wenn der entsprechende Parameter auf `true` gesetzt ist) angegeben werden. Im folgenden Beispiel wird ein BitmapData-Objekt mit orangefarbenem Hintergrund erstellt, der zu 50 % transparent ist:

```
var myBitmap:BitmapData = new BitmapData(150, 150, true, 0x80FF3300);
```

Wenn Sie ein neu erstelltes BitmapData-Objekt auf dem Bildschirm darstellen möchten, weisen Sie es einer Bitmap-Instanz zu oder schließen Sie es in eine Bitmap-Instanz ein. Dazu können Sie das BitmapData-Objekt entweder als Parameter für den Konstruktor des Bitmap-Objekts übergeben oder der `bitmapData`-Eigenschaft einer vorhandenen Bitmap-Instanz zuweisen. Darüber hinaus müssen Sie die Bitmap-Instanz zur Anzeigeliste hinzufügen, indem Sie die Methode `addChild()` oder `addChildAt()` des Anzeigeobjektcontainers aufrufen, der die Bitmap-Instanz enthält. Weitere Informationen zur Verwendung der Anzeigeliste finden Sie unter „Hinzufügen von Anzeigeobjekten zur Anzeigeliste“ auf Seite 170.

Im folgenden Beispiel wird ein BitmapData-Objekt mit roter Füllung erstellt und in einer Bitmap-Instanz angezeigt:

```
var myBitmapDataObject:BitmapData = new BitmapData(150, 150, false, 0xFF0000);
var myImage:Bitmap = new Bitmap(myBitmapDataObject);
addChild(myImage);
```

Bearbeiten von Pixelwerten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die BitmapData-Klasse enthält mehrere Methoden zum Bearbeiten von Pixeldatenwerten.

Bearbeiten einzelner Pixel

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Ändern der Darstellung eines Bitmapbilds auf Pixelebene müssen Sie zunächst die Farbwerte der Pixel in dem zu bearbeitenden Bereich abrufen. Die entsprechenden Pixelwerte können mithilfe der `getPixel()`-Methode gelesen werden.

Mit der `getPixel()`-Methode wird ein RGB-Wert für ein x- und y-Koordinatenpaar (Pixel) abgerufen, das als Parameter übergeben wird. Wenn die zu bearbeitenden Pixel Transparenzinformationen (Alphakanal) enthalten, muss die `getPixel32()`-Methode verwendet werden. Mit dieser Methode wird ebenfalls ein RGB-Wert abgerufen. Im Gegensatz zu `getPixel()` enthält der von `getPixel32()` zurückgegebene Wert jedoch zusätzliche Daten, mit denen der Wert des Alphakanals (Transparenz) des ausgewählten Pixels angegeben wird.

Wenn Sie dagegen lediglich die Farbe oder die Transparenz eines Pixels in einer Bitmap ändern möchten, können Sie die `setPixel()`-Methode oder die `setPixel32()`-Methode verwenden. Übergeben Sie zum Festlegen der Farbe eines Pixels die x- und die y-Koordinate sowie den Farbwert an eine dieser Methoden.

Im folgenden Beispiel wird mithilfe von `setPixel()` ein Kreuz auf einem grünen BitmapData-Hintergrund gezeichnet. Anschließend werden mit `getPixel()` der Farbwert des Pixels an der Koordinate (50, 50) abgerufen und der zurückgegebene Wert ausgegeben.

```
import flash.display.Bitmap;
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 100, false, 0x009900);

for (var i:uint = 0; i < 100; i++)
{
    var red:uint = 0xFF0000;
    myBitmapData.setPixel(50, i, red);
    myBitmapData.setPixel(i, 50, red);
}

var myBitmapImage:Bitmap = new Bitmap(myBitmapData);
addChild(myBitmapImage);

var pixelValue:uint = myBitmapData.getPixel(50, 50);
trace(pixelValue.toString(16));
```

Verwenden Sie die `getPixels()`-Methode, wenn Sie anstelle eines einzelnen Pixelwerts den Wert einer Pixelgruppe abrufen möchten. Mit dieser Methode wird ein Byte-Array eines rechteckigen Bereichs von Pixeldaten erstellt, das als Parameter übergeben wird. Alle Elemente des Byte-Arrays (d. h. die einzelnen Pixelwerte) sind vorzeichenlose 32-Bit-Ganzzahlen und nicht multiplizierte Farbwerte.

Wenn Sie im Gegensatz dazu den Wert einer Pixelgruppe ändern (oder festlegen) möchten, verwenden Sie die `setPixels()`-Methode. Bei dieser Methode müssen zwei Parameter (`rect` und `inputByteArray`) angegeben werden, die zur Ausgabe eines rechteckigen Bereichs (`rect`) von Pixeldaten (`inputByteArray`) kombiniert werden.

Beim Lesen (und Schreiben) von Daten in `inputByteArray` wird die `ByteArray.readUnsignedInt()`-Methode für alle Pixel im Array aufgerufen. Wenn das `inputByteArray` aus bestimmten Gründen kein vollständiges Rechteck mit Pixeldaten enthält, wird die Verarbeitung der Bilddaten an der entsprechenden Position abgebrochen.

Bedenken Sie dabei immer, dass beim Abrufen und Festlegen von Pixeldaten für das Byte-Array 32-Bit-Pixelwerte für Alpha, Rot, Grün und Blau (ARGB) angegeben werden müssen.

Im folgenden Beispiel wird mithilfe der Methoden `getPixels()` und `setPixels()` eine Gruppe von Pixeldaten von einem BitmapData-Objekt in ein anderes BitmapData-Objekt kopiert:

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.utils.ByteArray;
import flash.geom.Rectangle;

var bitmapDataObject1:BitmapData = new BitmapData(100, 100, false, 0x006666FF);
var bitmapDataObject2:BitmapData = new BitmapData(100, 100, false, 0x00FF0000);

var rect:Rectangle = new Rectangle(0, 0, 100, 100);
var bytes:ByteArray = bitmapDataObject1.getPixels(rect);

bytes.position = 0;
bitmapDataObject2.setPixels(rect, bytes);

var bitmapImage1:Bitmap = new Bitmap(bitmapDataObject1);
addChild(bitmapImage1);
var bitmapImage2:Bitmap = new Bitmap(bitmapDataObject2);
addChild(bitmapImage2);
bitmapImage2.x = 110;
```

Kollisionserkennung auf Pixelebene

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `BitmapData.hitTest()`-Methode wird eine Kollisionserkennung auf Pixelebene zwischen den Bitmapdaten und einem anderen Objekt oder Punkt durchgeführt.

Bei der `BitmapData.hitTest()`-Methode können die folgenden fünf Parameter angegeben werden:

- `firstPoint` (Point): Dieser Parameter verweist auf die Pixelposition der linken oberen Ecke des ersten BitmapData-Objekts, anhand dessen die Kollisionserkennung durchgeführt wird.
- `firstAlphaThreshold` (uint): Mit diesem Parameter wird der höchste Alphakanalwert angegeben, der bei der Kollisionserkennung als undurchsichtig ausgewertet wird.
- `secondObject` (Object): Dieser Parameter stellt den Kollisionsbereich dar. Beim `secondObject`-Objekt kann es sich um ein Rectangle-, Point-, Bitmap- oder BitmapData-Objekt handeln. Mit diesem Objekt wird der Kollisionsbereich angegeben, für den die Kollisionserkennung durchgeführt wird.
- `secondBitmapDataPoint` (Point): Mit diesem optionalen Parameter wird die Position eines Pixels im zweiten BitmapData-Objekt angegeben. Dieser Parameter wird nur verwendet, wenn der Wert des `secondObject`-Objekts ein BitmapData-Objekt ist. Der Standardwert ist `null`.
- `secondAlphaThreshold` (uint): Mit diesem optionalen Parameter wird der höchste Alphakanalwert angegeben, der im zweiten BitmapData-Objekt als undurchsichtig ausgewertet wird. Der Standardwert ist 1. Dieser Parameter wird nur verwendet, wenn der Wert von `secondObject` ein BitmapData-Objekt ist und beide BitmapData-Objekte Transparenzinformationen enthalten.

Denken Sie beim Durchführen der Kollisionserkennung für undurchsichtige Bilder daran, dass das Bild in ActionScript so verarbeitet wird, als handle es sich um vollständig undurchsichtige Rechtecke (oder Begrenzungsfelder). Beim Durchführen der Kollisionserkennung auf Pixelebene bei Bildern mit Transparenz müssen dagegen beide Bilder Transparenzinformationen enthalten. Darüber hinaus wird in ActionScript mithilfe der Parameter für den Alphaschwellenwert ermittelt, an welchem Punkt Pixel von transparent in undurchsichtig übergehen.

Im folgenden Beispiel werden drei Bitmapbilder erstellt. Anschließend wird eine Kollisionserkennung mit zwei verschiedenen Kollisionspunkten durchgeführt (ein Kollisionspunkt gibt „false“ zurück, der andere „true“):

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.geom.Point;

var bmd1:BitmapData = new BitmapData(100, 100, false, 0x000000FF);
var bmd2:BitmapData = new BitmapData(20, 20, false, 0x00FF3300);

var bm1:Bitmap = new Bitmap(bmd1);
this.addChild(bm1);

// Create a red square.
var redSquare1:Bitmap = new Bitmap(bmd2);
this.addChild(redSquare1);
redSquare1.x = 0;

// Create a second red square.
var redSquare2:Bitmap = new Bitmap(bmd2);
this.addChild(redSquare2);
redSquare2.x = 150;
redSquare2.y = 150;

// Define the point at the top-left corner of the bitmap.
var pt1:Point = new Point(0, 0);
// Define the point at the center of redSquare1.
var pt2:Point = new Point(20, 20);
// Define the point at the center of redSquare2.
var pt3:Point = new Point(160, 160);

trace(bmd1.hitTest(pt1, 0xFF, pt2)); // true
trace(bmd1.hitTest(pt1, 0xFF, pt3)); // false
```

Kopieren von Bitmapdaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um Bitmapdaten von einem Bild in ein anderes zu kopieren, können Sie verschiedene Methoden verwenden: `clone()`, `copyPixels()`, `copyChannel()`, `draw()` und `drawWithQuality()` (die `drawWithQuality`-Methode ist ab Flash Player 11.3 und ab AIR 3.3 verfügbar).

Mit der `clone()`-Methode können Sie Bitmapdaten von einem BitmapData-Objekt in ein anderes Objekt klonen. Beim Aufruf dieser Methode wird ein neues BitmapData-Objekt zurückgegeben, das ein exakter Klon der Instanz ist, von der es kopiert wurde.

Im folgenden Beispiel wird eine Kopie eines orangefarbenen (übergeordneten) Quadrats geklont und neben dem übergeordneten Originalquadrat positioniert:

```
import flash.display.Bitmap;
import flash.display.BitmapData;

var myParentSquareBitmap:BitmapData = new BitmapData(100, 100, false, 0x00ff3300);
var myClonedChild:BitmapData = myParentSquareBitmap.clone();

var myParentSquareContainer:Bitmap = new Bitmap(myParentSquareBitmap);
this.addChild(myParentSquareContainer);

var myClonedChildContainer:Bitmap = new Bitmap(myClonedChild);
this.addChild(myClonedChildContainer);
myClonedChildContainer.x = 110;
```

Mit der `copyPixels()`-Methode können Pixel schnell und einfach von einem BitmapData-Objekt in ein anderes kopiert werden. Bei dieser Methode wird ein rechteckiger Schnappschuss (definiert durch den `sourceRect`-Parameter) des Quellbilds erstellt und in einen anderen rechteckigen Bereich (mit der gleichen Größe) kopiert. Die Position des neuen „eingefügten" Rechtecks wird im `destPoint`-Parameter festgelegt.

Mit der `copyChannel()`-Methode wird ein vordefinierter Farbkanalwert (Alpha, Rot, Grün oder Blau) eines BitmapData-Quellobjekts in einen Kanal eines BitmapData-Zielobjekts kopiert. Der Aufruf dieser Methode wirkt sich nicht auf die anderen Kanäle im BitmapData-Zielobjekt aus.

Mit den `draw()`- und `drawWithQuality()`-Methoden wird der grafische Inhalt eines Sprite-Quellobjekts, MovieClip-Quellobjekts oder eines anderen Anzeigeobjekts in einer neuen Bitmap gezeichnet oder dargestellt. Mithilfe der Parameter `matrix`, `colorTransform`, `blendMode` und `clipRect` können Sie die Art und Weise ändern, auf die die neue Bitmap dargestellt wird. Diese Methode verwendet die Vektordarstellung in Flash Player und AIR, um die Daten zu generieren.

Beim Aufrufen von `draw()` oder `drawWithQuality()` wird das Quellobjekt (Sprite-Objekt, MovieClip-Objekt oder anderes Anzeigeobjekt) als erster Parameter übergeben, wie im Folgenden dargestellt:

```
myBitmap.draw(movieClip);
```

Wenn auf das Quellobjekt Transformationen (Farbe, Matrix usw.) angewendet wurden, nachdem es geladen wurde, werden diese Transformationen nicht in das neue Objekt kopiert. Wenn Sie die Transformationen in das neue Bitmapbild kopieren möchten, müssen Sie den Wert der `transform`-Eigenschaft des Originalobjekts in die `transform`-Eigenschaft des Bitmap-Objekts kopieren, in dem das neue BitmapData-Objekt verwendet wird.

Komprimieren von Bitmapdaten

Flash Player 11.3 und höher, AIR 3.3 und höher

Mit der `flash.display.BitmapData.encode()`-Methode können Sie Bitmapdaten nativ in eines der folgenden Bildformate komprimieren:

- **PNG** - Verwendet die PNG-Komprimierung, wahlweise mit schneller Komprimierung, bei der die Geschwindigkeit der Komprimierung wichtiger ist als die Dateigröße. Um die PNG-Komprimierung zu verwenden, übergeben Sie ein neues `flash.display.PNGEncoderOptions`-Objekt als zweiten Parameter der `BitmapData.encode()`-Methode.
- **JPEG** - Verwendet die JPEG-Komprimierung, wahlweise mit schneller Komprimierung. Um die JPEG-Komprimierung zu verwenden, übergeben Sie ein neues `flash.display.JPEGEncoderOptions`-Objekt als zweiten Parameter der `BitmapData.encode()`-Methode.

- **JPEGXR** - Verwendet die Komprimierung im Format JPEG Extended Range (XR), wahlweise unter Angabe der Einstellungen für Farbkanal, Verlust und Entropie. Um die JPEGXR-Komprimierung zu verwenden, übergeben Sie ein neues `flash.display.JPEGXREncoderOptions`-Objekt als zweiten Parameter der `BitmapData.encode()`-Methode.

Sie können diese Funktion für die Bildverarbeitung als Teil des Workflows bei einem Serverupload oder -download verwenden.

Im folgenden Beispielcodefragment wird ein BitmapData-Objekt unter Verwendung von `JPEGEncoderOptions` komprimiert:

```
// Compress a BitmapData object as a JPEG file.
var bitmapData:BitmapData = new BitmapData(640,480,false,0x00FF00);
var byteArray:ByteArray = new ByteArray();
bitmapData.encode(new Rectangle(0,0,640,480), new flash.display.JPEGEncoderOptions(),
byteArray);
```

Erstellen von Texturen mit Störungsfunktionen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie das Erscheinungsbild einer Bitmapgrafik ändern möchten, können Sie mithilfe der `noise()`-Methode oder der `perlinNoise()`-Methode einen Störungseffekt auf die Bitmap anwenden. Ein Störungseffekt kann mit den statischen Störungen bei einem auf keinen Sender eingestellten Fernseher verglichen werden.

Mithilfe der `noise()`-Methode können Sie einen Störungseffekt auf eine Bitmap anwenden. Mit dieser Methode wird ein zufällig gewählter Farbwert auf die Pixel in einem angegebenen Bereich eines Bitmapbilds angewendet.

Bei dieser Methode können die folgenden fünf Parameter angegeben werden:

- `randomSeed` (int): Der zufällig gewählte Anfangswert, mit dem das Muster festgelegt wird. Trotz des Namens werden mit gleichbleibendem Anfangswert immer die gleichen Ergebnisse generiert. Tatsächliche Zufallswerte können mithilfe der `Math.random()`-Methode generiert werden, mit der ein Zufallswert an diesen Parameter übergeben wird.
- `low` (uint): Dieser Parameter verweist auf den niedrigsten zu generierenden Wert für jedes Pixel (0 bis 255). Der Standardwert ist 0. Mit einem niedrigen Wert wird ein dunkles Störungsmuster generiert und mit einem höheren Wert ein helleres Muster.
- `high` (uint): Dieser Parameter verweist auf den höchsten zu generierenden Wert für jedes Pixel (0 bis 255). Der Standardwert ist 255. Mit einem niedrigen Wert wird ein dunkles Störungsmuster generiert und mit einem höheren Wert ein helleres Muster.
- `channelOptions` (uint): Mit diesem Parameter wird der Farbkanal des Bitmap-Objekts angegeben, auf den das Störungsmuster angewendet wird. Die Zahl kann eine beliebige Kombination der vier Farbkanäle für die ARGB-Werte sein. Der Standardwert ist 7.
- `grayScale` (Boolean): Wenn dieser Parameter auf `true` gesetzt ist, wird der `randomSeed`-Wert auf die Pixel des Bitmapbilds angewendet. Dabei wird alle Farbe aus dem Bild entfernt. Dieser Parameter wirkt sich nicht auf den Alphakanal aus. Der Standardwert lautet `false`.

Im folgenden Beispiel wird ein Bitmapbild erstellt, auf das ein blaues Störungsmuster angewendet wird:

```
package
{
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.BitmapDataChannel;

    public class BitmapNoise1 extends Sprite
    {
        public function BitmapNoise1()
        {
            var myBitmap:BitmapData = new BitmapData(250, 250,false, 0xff000000);
            myBitmap.noise(500, 0, 255, BitmapDataChannel.BLUE,false);
            var image:Bitmap = new Bitmap(myBitmap);
            addChild(image);
        }
    }
}
```

Wenn Sie eine naturgetreuere Textur erstellen möchten, verwenden Sie die `perlinNoise()`-Methode. Mit der `perlinNoise()`-Methode können realistische, naturgetreue Texturen erstellt werden, die sich optimal für Rauch, Wolken, Wasser, Feuer oder auch Explosionen eignen.

Die `perlinNoise()`-Methode wird mit einem Algorithmus generiert und belegt daher weniger Speicherplatz als bitmapbasierte Texturen. Vor allem auf älteren Computern kann sie sich jedoch auf die Prozessorauslastung auswirken, sodass der Inhalt beeinträchtigt wird und die Objekte auf dem Bildschirm mit einer unter der Bildrate liegenden Rate neu gezeichnet werden. Dies ist hauptsächlich auf die Gleitkommaberechnungen zurückzuführen, die für die Verarbeitung der Perlin-Störungsalgorithmen durchgeführt werden müssen.

Bei dieser Methode können die folgenden neun Parameter angegeben werden (die ersten sechs sind obligatorisch):

- `baseX` (Number): Legt den x-Wert (Größe) der erstellten Muster fest.
- `baseY` (Number): Legt den y-Wert (Größe) der erstellten Muster fest.
- `numOctaves` (uint): Anzahl der Oktaven bzw. der einzelnen Störungsfunktionen, die zur Erstellung der Störung kombiniert werden. Je höher die Anzahl der Oktaven, desto größer die Detailtreue der erstellten Bilder, desto länger jedoch auch die Verarbeitungszeit.
- `randomSeed` (int): Der zufällig gewählte Anfangswert entspricht dem in der `noise()`-Funktion. Tatsächliche Zufallswerte können mithilfe der `Math.random()`-Methode generiert werden, mit der ein Zufallswert an diesen Parameter übergeben wird.
- `stitch` (Boolean): Wenn dieser Parameter auf `true` gesetzt ist, werden die Übergänge des Bilds geglättet, um einen nahtlosen Texturenübergang für Füllmuster zu erstellen, mit denen Bitmaps gefüllt werden können.
- `fractalNoise` (Boolean): Dieser Parameter betrifft die Ränder der mit der Methode generierten Farbverläufe. Wenn der Parameter auf `true` gesetzt ist, wird eine fraktale Störung generiert, bei der die Ränder geglättet werden. Bei dem Wert `false` werden Turbulenzen generiert. In einem Bild mit Turbulenzen gibt es sichtbare Bruchstellen in den Farbverläufen, die sich gut für scharf abgegrenzte visuelle Effekte eignen, wie Flammen oder Wellen.
- `channelOptions` (uint): Der `channelOptions`-Parameter entspricht dem in der `noise()`-Methode. Mit diesem Parameter wird der Farbkanal (der Bitmap) angegeben, auf den das Störungsmuster angewendet wird. Die Zahl kann eine beliebige Kombination der vier Farbkanäle für die ARGB-Werte sein. Der Standardwert ist 7.
- `grayScale` (Boolean): Der `grayScale`-Parameter entspricht dem in der `noise()`-Methode. Wenn der Parameter auf `true` gesetzt ist, wird der `randomSeed`-Wert auf die Pixel des Bitmapbilds angewendet und dabei alle Farbe aus dem Bild entfernt. Der Standardwert lautet `false`.

- `offsets` (Array): Ein Array von Punkten, die dem x- und y-Offset jeder einzelnen Oktave entsprechen. Durch Ändern der Offset-Werte können Sie glatte Ebenenübergänge im Bild erzielen. Jeder Punkt im Offset-Array bezieht sich auf eine bestimmte Oktavenstörungsfunktion. Der Standardwert ist `null`.

Im folgenden Beispiel wird ein BitmapData-Objekt mit 150 x 150 Pixel erstellt, bei dem durch Aufrufen der `perlinNoise()`-Methode ein grüner und blauer Wolkeneffekt generiert wird:

```
package
{
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.BitmapDataChannel;

    public class BitmapNoise2 extends Sprite
    {
        public function BitmapNoise2()
        {
            var myBitmapDataObject:BitmapData =
                new BitmapData(150, 150, false, 0x00FF0000);

            var seed:Number = Math.floor(Math.random() * 100);
            var channels:uint = BitmapDataChannel.GREEN | BitmapDataChannel.BLUE
            myBitmapDataObject.perlinNoise(100, 80, 6, seed, false, true, channels, false, null);

            var myBitmap:Bitmap = new Bitmap(myBitmapDataObject);
            addChild(myBitmap);
        }
    }
}
```

Durchführen eines Bildlaufs in Bitmaps

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Stellen Sie sich vor, Sie haben eine Stadtplananwendung erstellt, bei der Sie die Ansicht stets aktualisieren müssen, wenn der Benutzer die Karte (selbst um nur wenige Pixel) verschiebt.

Eine Möglichkeit zum Realisieren dieser Funktionalität besteht darin, jedes Mal ein neues Bild mit der aktualisierten Ansicht des Stadtplans darzustellen, wenn der Benutzer den Stadtplan verschiebt. Alternativ dazu können Sie ein einzelnes großes Bild erstellen und die `scroll()`-Methode einsetzen.

Mit der `scroll()`-Methode wird eine auf dem Bildschirm angezeigte Bitmap kopiert und dann an einer neuen, versetzten Position eingefügt. Diese Position wird durch die Parameter `x` und `y` angegeben. Wenn sich ein Bereich der Bitmap außerhalb der Bühne befindet, ergibt dies den Effekt, dass das Bild verschoben wurde. In Kombination mit einer Timer-Funktion (oder einem `enterFrame`-Ereignis) können Sie das Bild animieren bzw. einen Bildlauf realisieren.

Im folgenden Beispiel wird das vorherige Beispiel der Perlin-Störung zugrunde gelegt und ein größeres Bitmapbild erstellt. Dabei werden 3/4 des Bilds außerhalb der Bühne dargestellt. Dann wird die `scroll()`-Methode in Verbindung mit dem Ereignis-Listener `enterFrame` angewendet, mit dem das Bild um ein Pixel diagonal nach unten versetzt wird. Diese Methode wird jedes Mal zu Beginn eines Einzelbildes aufgerufen. Als Ergebnis werden die Bereiche außerhalb des Bildes nach und nach auf der Bühne dargestellt, während das Bild nach unten verschoben wird.

```
import flash.display.Bitmap;
import flash.display.BitmapData;

var myBitmapDataObject:BitmapData = new BitmapData(1000, 1000, false, 0x00FF0000);
var seed:Number = Math.floor(Math.random() * 100);
var channels:uint = BitmapDataChannel.GREEN | BitmapDataChannel.BLUE;
myBitmapDataObject.perlinNoise(100, 80, 6, seed, false, true, channels, false, null);

var myBitmap:Bitmap = new Bitmap(myBitmapDataObject);
myBitmap.x = -750;
myBitmap.y = -750;
addChild(myBitmap);

addEventListener(Event.ENTER_FRAME, scrollBitmap);

function scrollBitmap(event:Event):void
{
    myBitmapDataObject.scroll(1, 1);
}
```

Nutzen von MIP-Mapping

Flash Player 9 und höher, Adobe AIR 1.0 und höher

MIP-Maps (auch als *MipMaps* bezeichnet) sind Bitmaps, die gruppiert und mit einer Textur verknüpft sind, um die Darstellungsqualität und -leistung zur Laufzeit zu erhöhen. Jedes Bitmapbild in der MIP-Map ist eine Version des Hauptbitmapbilds, allerdings mit weniger Details als das Hauptbild.

Beispiel: Sie haben eine MIP-Map, die ein Hauptbild mit der höchsten Qualität von 64 x 64 Pixel enthält. Bilder mit geringerer Qualität in der MIP-Map haben die Auflösungen 32 × 32, 16 × 16, 8 × 8, 4 × 4, 2 × 2 und 1 × 1 Pixel.

Texturstreaming ist die Möglichkeit, die Bitmap mit der niedrigsten Qualität zuerst zu laden und dann progressiv Bitmaps mit höherer Qualität zu laden. Da die Bitmaps mit geringerer Qualität kleiner sind, werden sie schneller geladen als das Hauptbild. Deshalb können die Benutzer der Anwendung bereits ein Bild sehen, bevor die Hauptbitmap mit der hohen Qualität geladen wird.

Flash Player 9.115.0 und höhere Versionen sowie AIR implementieren diese Technologie (das Verfahren wird als *Mip-Mapping* bezeichnet), indem von jeder Bitmap optimierte Versionen mit unterschiedlicher Skalierung (beginnend bei 50 %) erstellt werden.

Flash Player 11.3 und AIR 3.3 unterstützen Texturstreaming mit dem `streamingLevels`-Parameter der Methoden `Context3D.createCubeTexture()` und `Context3D.createTexture()`.

Mithilfe der Texturkomprimierung können Sie Texturbilder im komprimierten Format direkt auf der GPU speichern, wodurch GPU-Speicher und Speicherbrandbreite eingespart werden können. Normalerweise werden komprimierte Texturen offline komprimiert und im komprimierten Format auf die GPU geladen. Flash Player 11.4 und AIR 3.4 unterstützen jedoch die Texturkomprimierung zur Laufzeit. Dies ist in bestimmten Situationen hilfreich, zum Beispiel beim Rendern dynamischer Texturen aus Vektorgrafiken. Führen Sie die folgenden Schritte aus, um die Texturkomprimierung zur Laufzeit zu verwenden:

- Erstellen Sie das Texturobjekt, indem Sie die `Context3D.createTexture()`-Methode aufrufen und dabei entweder `flash.display3D.Context3DTextureFormat.COMPRESSED` oder `flash.display3D.Context3DTextureFormat.COMPRESSED_ALPHA` im dritten Parameter übergeben.

- Verwenden Sie die `flash.display3D.textures.Texture`-Instanz, die vom `createTexture()`-Aufruf zurückgegeben wird, um entweder `flash.display3D.textures.Texture.uploadFromBitmapData()` oder `flash.display3D.textures.Texture.uploadFromByteArray()` aufzurufen. Mit diesen Methoden wird die Textur in einem Schritt hochgeladen und komprimiert.

MIP-Maps werden für die folgenden Bitmap-Typen erstellt:

- Bitmaps (JPEG-, GIF- oder PNG-Dateien), die mithilfe der ActionScript 3.0 Loader-Klasse angezeigt werden
- Bitmaps in der Bibliothek eines Flash Professional-Dokuments
- BitmapData-Objekte
- Bitmaps, die mithilfe der ActionScript 2.0 `loadMovie()`-Funktion angezeigt werden

MIP-Maps werden nicht auf gefilterte Objekte oder in Bitmaps zwischengespeicherte Movieclips angewendet. MIP-Maps werden jedoch angewendet, wenn Bitmaptransformationen in einem gefilterten Anzeigeobjekt vorliegen, selbst wenn die Bitmap in maskiertem Inhalt enthalten ist.

Das MIP-Mapping erfolgt automatisch, es gibt jedoch einige Richtlinien, durch deren Einhaltung Sie sicherstellen können, dass Ihre Bilder diese Optimierung nutzen:

- Stellen Sie für die Videowiedergabe die `smoothing`-Eigenschaft für das Video-Objekt auf `true` (siehe Abschnitt über die Video-Klasse).
- Für Bitmaps muss die `smoothing`-Eigenschaft nicht auf `true` eingestellt werden, die Qualitätsverbesserungen sind jedoch deutlicher sichtbar, wenn Bitmaps geglättet werden.
- Verwenden Sie für zweidimensionale Bilder Bitmapgrößen, die durch 4 oder 8 teilbar sind (z. B. 640 x 128; dieser Wert kann wie folgt verringert werden: 320 x 64 > 160 x 32 > 80 x 16 > 40 x 8 > 20 x 4 > 10 x > 5 x 1).

  Verwenden Sie für dreidimensionale Texturen MIP-Maps, bei denen jedes Bild eine Auflösung hat, die eine Potenz von 2 ist (2^n). Beispiel: Das Hauptbild hat eine Auflösung von 1024 x 1024 Pixel. Die Bilder mit niedrigerer Auflösung in der MIP-Map haben dann eine Auflösung von 512 x 512, 256 x 256, 128 x 128 bis zu 1 x 1 Pixel bei einer Gesamtzahl von 11 Bildern in der MIP-Map.

  Beachten Sie, dass Mipmapping nicht bei Bitmap-Inhalten mit einer extremen Breite oder Höhe erfolgt.

Bitmap-Beispiel: Animated Spinning Moon

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Im Beispiel „Animated Spinning Moon“ werden anhand eines animierten, sich drehenden Monds Techniken für die Verwendung von Bitmap-Objekten und Bitmap-Bilddaten (BitmapData-Objekten) demonstriert. Mithilfe eines zweidimensionalen Bilds der Mondoberfläche als Ausgangsmaterial wird die Animation eines sich drehenden, kugelförmigen Monds erzeugt. Es werden folgende Techniken veranschaulicht:

- Laden externer Bilder und Zugreifen auf die Bildrohdaten
- Erstellen eines Animationseffekts durch wiederholtes Kopieren von Pixeln aus unterschiedlichen Teilen eines Quellbilds
- Erstellen eines Bitmapbilds durch Festlegen von Pixelwerten

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „Animated Spinning Moon“ befinden sich im Ordner „Samples/SpinningMoon“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
SpinningMoon.mxml oder SpinningMoon.fla	Die Hauptanwendungsdatei in Flex (MXML) oder Flash (FLA).
com/example/programmingas3/moon/MoonSphere.as	Die Klasse mit den Funktionen zum Laden, Anzeigen und Animieren des Monds.
moonMap.png	Bilddatei mit einem Foto der Mondoberfläche, die geladen und zum Erstellen des animierten, sich drehenden Monds verwendet wird.

Laden externer Bilder als Bitmap-Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die erste Teilaufgabe in diesem Beispiel ist das Laden einer externen Bilddatei, die in diesem Fall ein Foto der Mondoberfläche ist. Der Ladevorgang wird mit zwei Methoden der MoonSphere-Klasse durchgeführt: dem `MoonSphere()`-Konstruktor, mit dem der Ladevorgang eingeleitet wird, und der `imageLoadComplete()`-Methode, die aufgerufen wird, wenn das externe Bild vollständig geladen wurde.

Das Laden externer Bilder entspricht dem Laden externer SWF-Dateien. In beiden Fällen wird zum Durchführen des Ladevorgangs eine Instanz der flash.display.Loader-Klasse verwendet. Der Code in der `MoonSphere()`-Methode, mit dem das Laden des Bilds gestartet wird, lautet wie folgt:

```
var imageLoader:Loader = new Loader();
imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoadComplete);
imageLoader.load(new URLRequest("moonMap.png"));
```

In der ersten Zeile wird die Loader-Instanz mit dem Namen `imageLoader` deklariert. In der dritten Zeile wird dann der Ladevorgang tatsächlich gestartet, indem die `load()`-Methode des Loader-Objekts aufgerufen und eine URLRequest-Instanz übergeben wird, die die URL des zu ladenden Bilds enthält. In der zweiten Zeile wird der Ereignis-Listener festgelegt, der ausgelöst wird, wenn das Bild vollständig geladen wurde. Beachten Sie, dass die `addEventListener()`-Methode nicht für die Loader-Instanz selbst aufgerufen wird, sondern für die `contentLoaderInfo`-Eigenschaft des Loader-Objekts. Die Loader-Instanz selbst löst keine Ereignisse aus, die sich auf die zu ladenden Inhalte beziehen. Die `contentLoaderInfo`-Eigenschaft der Instanz enthält jedoch einen Verweis auf das LoaderInfo-Objekt, das den in das Loader-Objekt zu ladenden Inhalten zugeordnet ist (in diesem Fall das externe Bild). Dieses LoaderInfo-Objekt enthält mehrere Ereignisse mit Bezug auf den Fortschritt und den Abschluss des Ladevorgangs externer Inhalte, u. a. das `complete`-Ereignis (`Event.COMPLETE`), das einen Aufruf der `imageLoadComplete()`-Methode auslöst, wenn das Bild vollständig geladen wurde.

Während das Starten des Ladens externer Bilder einen wichtigen Teil des Gesamtvorgangs darstellt, ist es genau so wichtig, sich über die folgenden Schritte im Klaren zu sein, nachdem das Laden abgeschlossen ist. Wie im zuvor abgebildeten Code dargestellt, wird die `imageLoadComplete()`-Funktion aufgerufen, wenn das Bild vollständig geladen ist. In dieser Funktion erfolgen verschiedene Manipulationen der geladenen Bilddaten, die in den folgenden Abschnitten beschrieben werden. Zum Verwenden der Bilddaten muss jedoch zunächst auf sie zugegriffen werden. Beim Verwenden eines Loader-Objekts zum Laden eines externen Bilds wird das geladene Bild zu einer Bitmap-Instanz, die dem Loader-Objekt als untergeordnetes Anzeigeobjekt zugeordnet wird. In diesem Fall steht die Loader-Instanz in der Ereignis-Listener-Methode als Teil des Ereignisobjekts zur Verfügung, das der Methode als Parameter übergeben wird. Es folgen die ersten Zeilen der `imageLoadComplete()`-Methode:

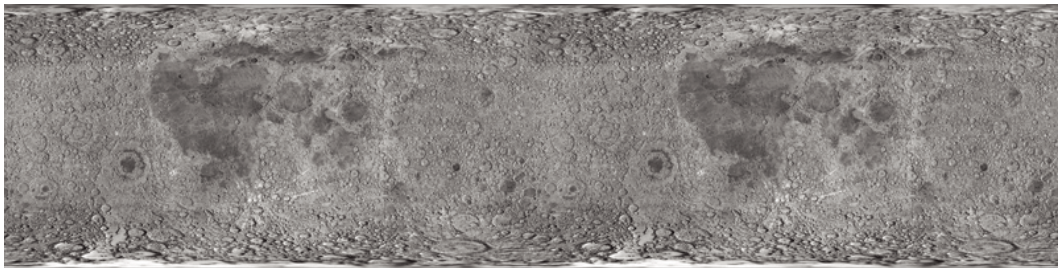
```
private function imageLoadComplete(event:Event):void
{
    textureMap = event.target.content.bitmapData;
    ...
}
```

Beachten Sie, dass der Ereignisobjektparameter die Bezeichnung `event` trägt und eine Instanz der Event-Klasse ist. Jede Instanz der Event-Klasse hat eine `target`-Eigenschaft, die auf das Objekt verweist, das das Ereignis ausgelöst hat (in diesem Fall die LoaderInfo-Instanz, für die wie zuvor beschrieben die `addEventListener()`-Methode aufgerufen wurde). Das LoaderInfo-Objekt hat wiederum eine `content`-Eigenschaft, die (nach Abschluss des Ladevorgangs) die Bitmap-Instanz mit dem geladenen Bitmapbild enthält. Wenn Sie das Bild direkt auf dem Bildschirm anzeigen möchten, können Sie diese Bitmap-Instanz (`event.target.content`) einem Anzeigeobjektcontainer zuweisen. (Sie können auch das Loader-Objekt einem Anzeigeobjektcontainer zuweisen). In diesem Beispiel werden die geladenen Inhalte jedoch als Quelle für Bildrohdaten verwendet und nicht direkt auf dem Bildschirm angezeigt. Daher wird in der ersten Zeile der `imageLoadComplete()`-Methode die `bitmapData`-Eigenschaft der geladenen Bitmap-Instanz (`event.target.content.bitmapData`) gelesen und in der Instanzvariablen `textureMap` gespeichert. Diese wird als Bilddatenquelle verwendet, um die Animation des sich drehenden Monds zu erzeugen. Dies wird als Nächstes beschrieben.

Erstellen eines Animationseffekts durch Kopieren von Pixeln

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine einfache Definition von Animation ist die Illusion von Bewegung oder Abwechslung durch das Ändern eines Bilds über einen bestimmten Zeitraum. In diesem Beispiel besteht das Ziel darin, die Illusion eines kugelförmigen Monds zu erzeugen, der sich um seine vertikale Achse dreht. Für den Zweck dieser Animation kann jedoch der Aspekt einer sphärischen Verzerrung vernachlässigt werden. Betrachten Sie das Bild, das geladen und als Quelle der Mondbilddaten verwendet wird:



Wie Sie sehen, sind in dem Bild keine kugelförmigen Körper abgebildet. Vielmehr handelt es sich um ein rechteckiges Foto der Mondoberfläche. Da das Foto eine Aufnahme vom Mondäquator darstellt, sind die Bildbereiche am oberen und unteren Rand lang gezogen und verzerrt. Um die Bildverzerrung zu entfernen und das Bild kugelförmig erscheinen zu lassen, wird ein Verschiebungsmatrixfilter eingesetzt. Dies wird an späterer Stelle beschrieben. Da dieses Quellbild ein Rechteck ist, genügt es zum Erzeugen der Illusion einer sich drehenden Kugel, das Foto der Mondoberfläche horizontal zu verschieben.

Beachten Sie, dass das Bild eigentlich aus zwei nebeneinandergelegten Kopien des Mondoberflächenfotos besteht. Dieses Bild ist das Quellbild, aus dem wiederholt Bilddaten kopiert werden, um den Anschein von Bewegung zu erwecken. Durch das Aneinanderfügen zweier Kopien des Bilds ist es einfacher, einen ununterbrochenen Bildlaufeffekt zu erzielen. Zum besseren Verständnis wird der Animationsvorgang nun schrittweise abgehandelt.

Der Animationsvorgang wird mit zwei verschiedenen ActionScript-Objekten durchgeführt. Zunächst wird das geladene Quellbild verwendet, das im Code durch die BitmapData-Instanz mit dem Namen `textureMap` dargestellt wird. Wie zuvor beschrieben, wird `textureMap` mithilfe des folgenden Codes mit Bilddaten gefüllt, wenn der Ladevorgang des externen Bilds abgeschlossen ist:

```
textureMap = event.target.content.bitmapData;
```

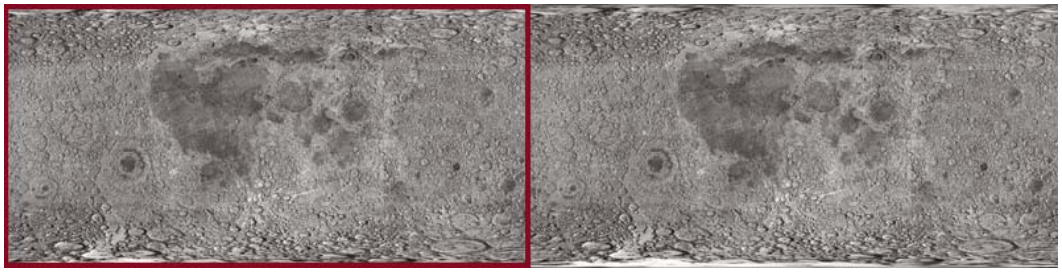
Der Inhalt von `textureMap` ist das rechteckige Mondbild. Zusätzlich wird im Code zum Erzeugen des animierten Rotationseffekts eine Bitmap-Instanz mit dem Namen `sphere` verwendet, bei der es sich um das eigentliche Anzeigeobjekt handelt, mit dem das Mondbild auf dem Bildschirm angezeigt wird. Wie `textureMap` wird auch das `sphere`-Objekt mithilfe der ursprünglichen Bilddaten in der `imageLoadComplete()`-Methode erzeugt und gefüllt. Dies erfolgt mit dem folgenden Code:

```
sphere = new Bitmap();
sphere.bitmapData = new BitmapData(textureMap.width / 2, textureMap.height);
sphere.bitmapData.copyPixels(textureMap,
                        new Rectangle(0, 0, sphere.width, sphere.height),
                        new Point(0, 0));
```

Wie aus dem Code ersichtlich, wird `sphere` instanziiert. Die `bitmapData`-Eigenschaft der Instanz (die Bildrohdaten, die von `sphere` angezeigt werden) wird mit derselben Höhe und der halben Breite von `textureMap` angelegt. Anders ausgedrückt, hat der Inhalt von `sphere` die Größe eines Mondfotos (da das Bild in `textureMap` aus zwei nebeneinandergelegten Mondfotos besteht). Als Nächstes wird die `bitmapData`-Eigenschaft mithilfe der `copyPixels()`-Methode mit Bilddaten gefüllt. Mit den Parametern im Aufruf der `copyPixels()`-Methode werden mehrere Informationen übergeben:

- Der erste Parameter gibt an, dass die Bilddaten aus `textureMap` kopiert werden.
- Der zweite Parameter, eine neue Rectangle-Instanz, legt fest, aus welchem Teil von `textureMap` der Bildausschnitt kopiert werden soll. In diesem Fall handelt es sich um ein Rechteck, das an der linken oberen Ecke von `textureMap` beginnt (angegeben durch die ersten beiden `Rectangle()`-Parameter `0, 0`). Die Breite und Höhe des rechteckigen Bildausschnitts stimmen mit den Eigenschaften `width` und `height` von `sphere` überein.
- Der dritte Parameter, eine neue Point-Instanz mit dem x-Wert und dem y-Wert `0`, definiert das Ziel der Pixeldaten, in diesem Fall die linke obere Ecke (0, 0) von `sphere.bitmapData`.

Mit dem Code werden die in der folgenden Abbildung umrandeten Pixel aus `textureMap` kopiert und in `sphere` eingefügt. Anders ausgedrückt, ist der BitmapData-Inhalt von `sphere` der im Folgenden markierte Teil von `textureMap`:



Dabei darf jedoch nicht vergessen werden, dass es sich hierbei nur um den Anfangszustand von `sphere` handelt, d. h. um das erste Bild, das in `sphere` eingefügt wird.

Nachdem nun das Quellbild geladen und `sphere` erstellt wurde, besteht die letzte Aufgabe der `imageLoadComplete()`-Methode darin, die Animation einzurichten. Die Animation wird von einer Timer-Instanz mit dem Namen `rotationTimer` gesteuert, die mit dem folgenden Code erstellt und gestartet wird:

```
var rotationTimer:Timer = new Timer(15);
rotationTimer.addEventListener(TimerEvent.TIMER, rotateMoon);
rotationTimer.start();
```

Im Code wird zunächst die Timer-Instanz mit dem Namen `rotationTimer` erstellt. Mit dem an den `Timer()`-Konstruktor übergebenen Parameter wird angegeben, dass `rotationTimer` das zugehörige `timer`-Ereignis alle 15 illisekunden auslösen soll. Anschließend wird die `addEventListener()`-Methode aufgerufen und festgelegt, dass beim Auftreten des `timer`-Ereignisses (`TimerEvent.TIMER`) die Methode `rotateMoon()` aufgerufen wird. Schließlich wird der Timer dann durch Aufrufen der zugehörigen `start()`-Methode gestartet.

Aufgrund der Parameter für `rotationTimer` ruft Flash Player im Abstand von etwa 15 Millisekunden die `rotateMoon()`-Methode der MoonSphere-Klasse auf. Auf diese Weise wird die Animation des Monds realisiert. Der Quellcode der `rotateMoon()`-Methode lautet wie folgt:

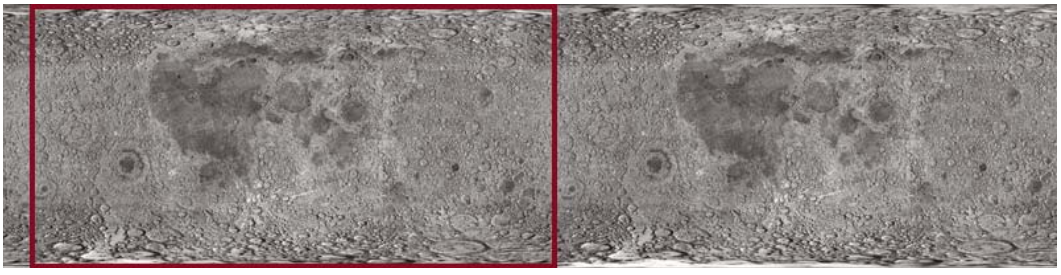
```
private function rotateMoon(event:TimerEvent):void
{
    sourceX += 1;
    if (sourceX > textureMap.width / 2)
    {
        sourceX = 0;
    }

    sphere.Data.copyPixels(textureMap,
                                   new Rectangle(sourceX, 0, sphere.width, sphere.height),
                                   new Point(0, 0));

    event.updateAfterEvent();
}
```

Mit diesem Code werden drei Aufgaben ausgeführt:

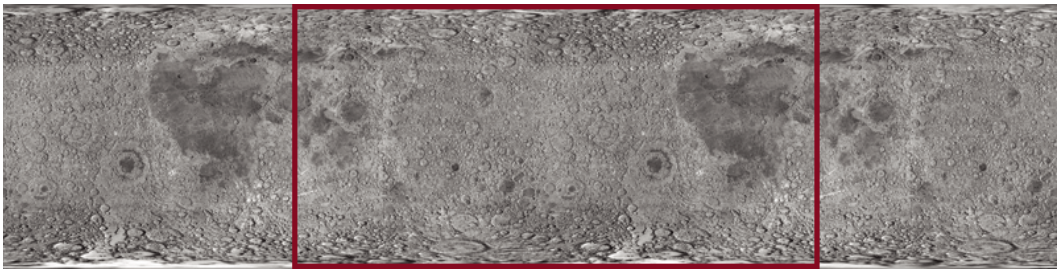
1 Der Wert der Variablen `sourceX` (anfangs auf 0 gesetzt) wird um 1 erhöht.

```
sourceX += 1;
```

Die Variable `sourceX` wird verwendet, um die Position in `textureMap` festzulegen, ab der die Pixel kopiert und in `sphere` eingefügt werden. Der Code hat daher den Effekt, dass das auf `textureMap` angewendete Rechteck um ein Pixel nach rechts verschoben wird. In der grafischen Darstellung bedeutet dies, dass das Quellrechteck nach mehreren Animationszyklen um mehrere Pixel nach rechts verschoben ist:

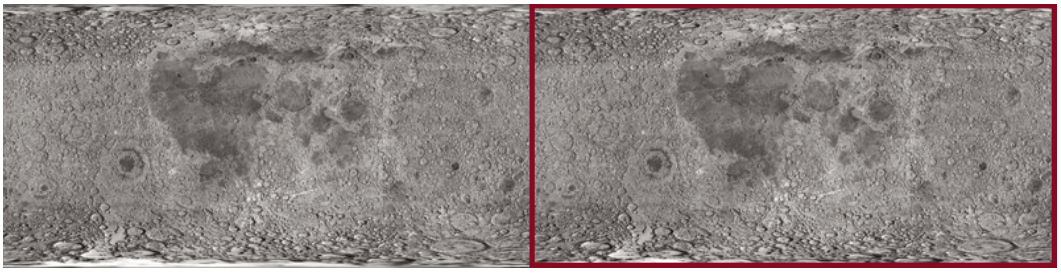


Nach einigen weiteren Zyklen hat sich das Rechteck noch weiter bewegt:



Diese schrittweise, kontinuierliche Verschiebung der Position, ab der die Pixel kopiert werden, ist der Schlüssel zur Animation. Durch das langsame und stetige Verschieben der Quellposition nach rechts scheint sich das auf dem Bildschirm in `sphere` angezeigte Bild kontinuierlich nach links zu bewegen. Aus diesem Grund muss das Quellbild (`textureMap`) aus zwei Kopien des Mondoberflächenfotos bestehen. Da sich das Rechteck kontinuierlich nach rechts bewegt, befindet es sich die meiste Zeit nicht über nur einem Mondfoto, sondern überlappt die beiden Fotos.

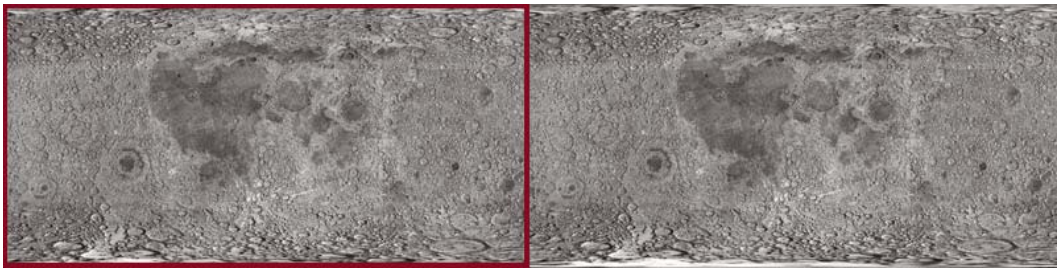
2 Es gibt jedoch ein Problem mit dem sich langsam nach rechts verschiebenden Rechteck. Über kurz oder lang erreicht das Rechteck den rechten Rand von `textureMap`. Gleichzeitig können keine weiteren Pixel des Mondfotos kopiert und in `sphere` eingefügt werden.



Dieses Problem wird mit den nächsten Codezeilen gelöst:

```
if (sourceX >= textureMap.width / 2)
{
    sourceX = 0;
}
```

Im Code wird überprüft, ob `sourceX` (der linke Rand des Rechtecks) die Mitte von `textureMap` erreicht hat. Wenn dies der Fall ist, wird `sourceX` wieder auf 0 gesetzt und so zurück zum linken Rand von `textureMap` bewegt. Damit beginnt der Zyklus von vorn:



3 Nachdem der passende Wert für `sourceX` berechnet ist, besteht der letzte Schritt zum Erzeugen der Animation darin, die Pixel des neuen Quellrechtecks auch tatsächlich zu kopieren und in `sphere` einzufügen. Der dazu verwendete Code ähnelt sehr dem Code, mit dem `sphere` zu Beginn gefüllt wurde (weiter oben beschrieben). Der einzige Unterschied besteht darin, dass nun beim Aufrufen des Konstruktors `new Rectangle()` der linke Rand des Rechtecks auf `sourceX` gesetzt wird:

```
sphere.bitmapData.copyPixels(textureMap,
                             new Rectangle(sourceX, 0, sphere.width, sphere.height),
                             new Point(0, 0));
```

Rufen Sie sich in Erinnerung, dass dieser Code alle 15 illisekunden aufgerufen wird. Während sich die Position des Quellrechtecks kontinuierlich ändert und die kopierten Pixel in `sphere` eingefügt werden, wird auf dem Bildschirm der Eindruck erzeugt, dass sich das in `sphere` dargestellte Mondfoto ständig bewegt. Anders ausgedrückt, es entsteht der Eindruck, dass der Mond sich dreht.

Erzeugen des kugelförmigen Erscheinungsbilds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Natürlich ist der Mond kein Rechteck, sondern eine Kugel. Daher muss im Beispiel das rechteckige und fortlaufend animierte Foto der Mondoberfläche in eine Kugel umgewandelt werden. Hierzu müssen zwei Schritte ausgeführt werden: Mit einer Maske werden alle Bildinhalte außerhalb eines kreisförmigen Bereichs des Mondoberflächenfotos ausgeblendet. Mit einem Verschiebungsmatrixfilter wird dann die Darstellung des Mondfotos verzerrt, um es dreidimensional erscheinen zu lassen.

Zunächst wird eine kreisförmige Maske verwendet, um alle Inhalte des MoonSphere-Objekts außer der durch den Filter erzeugten Kugel auszublenden. Mit dem folgenden Code wird die Maske als Shape-Instanz erstellt und auf die MoonSphere-Instanz angewendet:

```
moonMask = new Shape();
moonMask.graphics.beginFill(0);
moonMask.graphics.drawCircle(0, 0, radius);
this.addChild(moonMask);
this.mask = moonMask;
```

Beachten Sie, dass die Maske über die geerbte `mask`-Eigenschaft direkt auf die MoonSphere-Instanz angewendet werden kann, da es sich bei dieser Instanz um ein Anzeigeobjekt handelt.

Um einen realistisch wirkenden Effekt einer sich drehenden Kugel zu erzeugen, ist es nicht damit getan, Teile des Fotos durch eine kreisförmige Maske auszublenden. Aufgrund der Art und Weise, wie das Foto der Mondoberfläche aufgenommen wurde, sind die Abmessungen nicht proportional. Die Bildteile am oberen und unteren Bildrand sind stärker verzerrt und gestreckt als die Bereiche am Äquator. Um das Mondfoto dreidimensional wirken zu lassen, wird ein Verschiebungsmatrixfilter verwendet.

Bei einem Verschiebungsmatrixfilter handelt es sich um einen Filtertyp, der zum Verzerren von Bildern eingesetzt wird. In diesem Fall wird das Mondfoto „verzerrt“, damit es realistischer wirkt. Dazu werden der obere und untere Bildteil horizontal gestaucht, der Mittelteil bleibt jedoch unverändert. Unter der Voraussetzung, dass der Filter auf einen quadratförmigen Teil des Fotos angewendet wird, entsteht durch Stauchen des oberen und unteren Bildteils ohne Einbeziehung der Bildmitte aus dem Quadrat ein Kreis. Ein Nebeneffekt der Animation dieses verzerrten Bilds besteht darin, dass sich die Bildmitte jeweils um einen größeren Pixelabstand zu bewegen scheint als die Bereiche am oberen und unteren Bildrand. Dies erzeugt die Illusion, dass es sich bei dem Kreis in Wirklichkeit um ein dreidimensionales Objekt (eine Kugel) handelt.

Der folgende Code wird verwendet, um den Verschiebungsmatrixfilter mit dem Namen `displaceFilter` zu erstellen:

```
var displaceFilter:DisplacementMapFilter;
displaceFilter = new DisplacementMapFilter(fisheyeLens,
                                new Point(radius, 0),
                                BitmapDataChannel.RED,
                                BitmapDataChannel.GREEN,
                                radius, 0);
```

Der erste Parameter `fisheyeLens` wird als Matrixbild bezeichnet. In diesem Fall handelt es sich um ein im Programmcode erstelltes BitmapData-Objekt. Die Erstellung dieses Bilds wird unter „Erstellen eines Bitmapbilds durch Festlegen von Pixelwerten“ auf Seite 279 erläutert. Die anderen Parameter beschreiben die Position im gefilterten Bild, ab der der Filter angewendet werden soll, sowie die zum Steuern des Verschiebungseffekts verwendeten Farbkanäle und in welchem Ausmaß sie sich auf die Verschiebung auswirken. Nachdem der Verschiebungsmatrixfilter erstellt wurde, wird er noch in der `imageLoadComplete()`-Methode auf `sphere` angewendet:

```
sphere.filters = [displaceFilter];
```

Das endgültige Bild (Maske und Verschiebungsmatrixfilter angewendet) sieht wie folgt aus:

Mit jedem neuen Zyklus der Animation des sich drehenden Monds wird der Inhalt der BitmapData-Eigenschaft von „sphere“ mit einem neuen Ausschnitt der Quellbilddaten überschrieben. Der Filter muss jedoch nicht jedes Mal erneut angewendet werden. Der Grund hierfür ist, dass der Filter nicht auf die Bitmap-Daten (die Pixelrohdaten), sondern auf die Bitmap-Instanz (das Anzeigeobjekt) angewendet wird. Beachten Sie, dass die Bitmap-Instanz nicht die tatsächlichen Bitmap-Daten enthält. Sie ist das Anzeigeobjekt, mit dem die Bitmap-Daten auf dem Bildschirm dargestellt werden. Zur Veranschaulichung: Eine Bitmap-Instanz ist wie ein Diaprojektor, mit dem Fotos auf einer Leinwand dargestellt werden, und ein BitmapData-Objekt ist das eigentliche Dia, das mit dem Diaprojektor projiziert werden kann. Filter können direkt auf ein BitmapData-Objekt angewendet werden. Dies kann damit verglichen werden, dass direkt auf das Dia gezeichnet wird, um das Bild zu verändern. Filter können jedoch auch auf beliebige Anzeigeobjekte (einschließlich Bitmap-Instanzen) angewendet werden. Dies ist vergleichbar mit dem Anbringen eines Filters vor dem Projektorobjektiv, um das auf der Leinwand dargestellte Bild zu verzerren (ohne überhaupt das Original-Dia zu verändern). Da auf die Bitmap-Rohdaten mithilfe der bitmapData-Eigenschaft einer Bitmap-Instanz zugegriffen werden kann, können Filter direkt auf die Bitmap-Rohdaten angewendet werden. In diesem Fall ist es jedoch sinnvoll, den Filter nicht auf die Bitmap-Daten, sondern auf das Bitmap-Anzeigeobjekt anzuwenden.

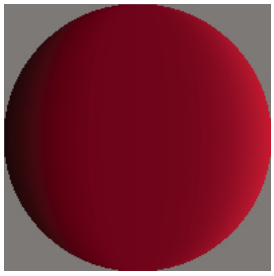
Weitere Informationen zum Verwenden von Verschiebungsmatrixfiltern in ActionScript finden Sie unter „Anwenden von Filtern auf Anzeigeobjekte“ auf Seite 286.

Erstellen eines Bitmapbilds durch Festlegen von Pixelwerten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein wichtiger Aspekt beim Anwenden von Verschiebungsmatrixfiltern ist, dass eigentlich zwei Bilder verwendet werden. Das eine Bild (das Quellbild) wird tatsächlich durch den Filter verändert. In diesem Beispiel ist das Quellbild die Bitmap-Instanz mit dem Namen `sphere`. Das andere vom Filter verwendete Bild wird als Matrixbild bezeichnet. Das Matrixbild wird nicht auf dem Bildschirm angezeigt. Stattdessen werden die Farben der einzelnen Pixel als Eingabe der Verschiebungsfunktion verwendet. Die Farbe des Pixels an einer bestimmten (x, y)-Koordinate im Matrixbild legt fest, welche Verschiebung (der physischen Position) auf das entsprechende Pixel an dieser (x, y)-Koordinate im Quellbild angewendet wird.

Daher wird für dieses Beispiel ein geeignetes Matrixbild benötigt, um mit dem Verschiebungsmatrixfilter einen Kugeleffekt zu erzeugen. Dieses Matrixbild hat einen grauen Hintergrund und stellt einen Kreis dar, der mit dem horizontalen Farbverlauf einer einzigen Farbe (Rot, von dunkel zu hell) gefüllt ist, wie im Folgenden abgebildet:



Da in diesem Beispiel nur ein Matrixbild und ein Filter eingesetzt werden, wird das Matrixbild nur einmal (in der `imageLoadComplete()`-Methode, d. h. nach Abschluss des Ladevorgangs des externen Bilds) erzeugt. Das Matrixbild mit dem Namen `fisheyeLens` wird durch Aufrufen der `createFisheyeMap()`-Methode der MoonSphere-Klasse erstellt:

```
var fisheyeLens:BitmapData = createFisheyeMap(radius);
```

Innerhalb der `createFisheyeMap()`-Methode wird das Matrixbild Pixel für Pixel mithilfe der `setPixel()`-Methode der BitmapData-Klasse gezeichnet. Der vollständige Code der `createFisheyeMap()`-Methode ist im Folgenden aufgeführt. Im Anschluss daran finden Sie eine schrittweise Erläuterung der Funktionsweise.

```
private function createFisheyeMap(radius:int):BitmapData
{
    var diameter:int = 2 * radius;

    var result:BitmapData = new BitmapData(diameter,
                                           diameter,
                                           false,
                                           0x808080);

    // Loop through the pixels in the image one by one
    for (var i:int = 0; i < diameter; i++)
    {
        for (var j:int = 0; j < diameter; j++)
        {
            // Calculate the x and y distances of this pixel from
            // the center of the circle (as a percentage of the radius).
            var pctX:Number = (i - radius) / radius;
            var pctY:Number = (j - radius) / radius;

            // Calculate the linear distance of this pixel from
            // the center of the circle (as a percentage of the radius).
            var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);

            // If the current pixel is inside the circle,
```

```
            // set its color.
            if (pctDistance < 1)
            {
                // Calculate the appropriate color depending on the
                // distance of this pixel from the center of the circle.
                var red:int;
                var green:int;
                var blue:int;
                var rgb:uint;
                red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
                green = 0;
                blue = 0;
                rgb = (red << 16 | green << 8 | blue);
                // Set the pixel to the calculated color.
                result.setPixel(i, j, rgb);
            }
        }
    }
    return result;
}
```

Beim Aufrufen der Methode wird dieser zunächst ein Parameter (`radius`) übergeben, der den Radius des zu erstellenden kreisförmigen Bilds angibt. Anschließend wird im Code das BitmapData-Objekt erstellt, in dem der Kreis gezeichnet werden soll. Dieses Objekt mit dem Namen `result` wird schließlich als Rückgabewert der Methode zurückgegeben. Wie im folgenden Codeausschnitt dargestellt, wird die BitmapData-Instanz `result` mit den folgenden Werten erstellt: Breite und Höhe entsprechen dem Kreisdurchmesser, keine Transparenz (`false` im dritten Parameter) und Vorfüllung mit der Farbe `0x808080` (mittelgrau):

```
var result:BitmapData = new BitmapData(diameter,
                                       diameter,
                                       false,
                                       0x808080);
```

Anschließend werden im Code zwei Schleifen verwendet, damit alle Pixel des Bilds einzeln durchlaufen werden. In der äußeren Schleife werden alle Bildspalten von links nach rechts durchlaufen (die Variable `i` enthält die horizontale Position des jeweils zu bearbeitenden Pixels). In der inneren Schleife werden alle Pixel der aktiven Spalte von oben nach unten durchlaufen (die Variable `j` enthält die vertikale Position des jeweils zu bearbeitenden Pixels). Der Code für die Schleifen ist im Folgenden dargestellt (der Inhalt der inneren Schleife wurde weggelassen):

```
for (var i:int = 0; i < diameter; i++)
{
    for (var j:int = 0; j < diameter; j++)
    {
        ...
    }
}
```

Während in den Schleifen alle Pixel einzeln durchlaufen werden, wird für jedes Pixel ein Wert (der Farbwert dieses Pixels im Matrixbild) berechnet. Dieser Vorgang umfasst vier Schritte:

1. Im Code wird für das aktuelle Pixel der Abstand vom Kreismittelpunkt auf der x-Achse berechnet (`i - radius`). Dieser Wert wird durch den Radius dividiert, um den Prozentwert (im Verhältnis zum Radius) und nicht den absoluten Abstand (`(i - radius) / radius`) zu erhalten. Dieser Prozentwert wird in einer Variablen mit dem Namen `pctX` gespeichert. Der entsprechende Wert für die y-Achse wird ebenfalls berechnet und in der Variablen `pctY` gespeichert, wie im folgenden Code dargestellt:

```
var pctX:Number = (i - radius) / radius;
var pctY:Number = (j - radius) / radius;
```

2 Mithilfe einer trigonometrischen Standardformel, dem Satz des Pythagoras, wird aus `pctX` und `pctY` der lineare Abstand zwischen dem Kreismittelpunkt und dem aktuellen Punkt berechnet. Dieser Wert wird in einer Variablen mit dem Namen `pctDistance` gespeichert, wie im Folgenden dargestellt:

```
var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);
```

3 Anschließend wird überprüft, ob der prozentuale Abstand kleiner als 1 ist (100 % des Radius, d. h. das Pixel befindet sich innerhalb des Kreisradius). Wenn sich das Pixel innerhalb des Kreises befindet, wird ihm ein berechneter Wert (an dieser Stelle weggelassen, jedoch in Schritt 4 beschrieben) zugewiesen. Andernfalls wird der Standardfarbwert des Pixels für Mittelgrau nicht geändert:

```
if (pctDistance < 1)
{
    ...
}
```

4 Für die Pixel innerhalb des Kreises wird ein Farbwert berechnet. Am Ende weist der Kreis einen roten Farbverlauf auf, der sich von Schwarz (0 % Rot) am linken Kreisrand bis zu Hellrot (100 %) am rechten Kreisrand erstreckt. Der Farbwert wird ursprünglich in drei Teilkomponenten (Rot, Grün und Blau) berechnet, wie im Folgenden dargestellt:

```
red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
green = 0;
blue = 0;
```

Beachten Sie, dass nur der Rotanteil der Farbe (die Variable `red`) tatsächlich einen Wert enthält. Die Werte für Grün und Blau (die Variablen `green` und `blue`) sind hier zum besseren Verständnis ebenfalls aufgeführt, können jedoch weggelassen werden. Da der Zweck dieser Methode darin besteht, einen Kreis mit einem roten Farbverlauf zu erzeugen, werden keine Werte für Grün und Blau benötigt.

Nachdem die drei Farbwertkomponenten festgelegt sind, werden sie mithilfe eines Bitverschiebungs-Standardalgorithmus in einem einzigen ganzzahligen Farbwert kombiniert, wie im folgenden Code dargestellt:

```
rgb = (red << 16 | green << 8 | blue);
```

Schließlich wird der berechnete Farbwert dann mit der `setPixel()`-Methode des BitmapData-Objekts `result` dem aktuellen Pixel zugewiesen:

```
result.setPixel(i, j, rgb);
```

Asynchrone Dekodierung von Bitmapbildern

Flash Player 11 und höher, Adobe AIR 2.6 und höher

Bei der Arbeit mit Bitmapbildern können Sie die Bitmaps asynchron dekodieren und laden, um die wahrgenommene Leistung Ihrer Anwendung zu verbessern. In vielen Fällen kann das asynchrone Dekodieren der Bitmap genauso viel Zeit in Anspruch nehmen wie das synchrone Dekodieren. Die Bitmap wird jedoch in einem separaten Thread dekodiert, bevor das zugehörige `Loader`-Objekt das `COMPLETE`-Ereignis sendet. Deshalb können Sie größere Bilder asynchron dekodieren, nachdem Sie sie geladen haben.

Über die `ImageDecodingPolicy`-Klasse im `flash.system`-Paket können Sie das Schema zum Laden der Bitmaps angeben. Das standardmäßige Ladeschema ist synchron.

Richtlinie zum Dekodieren von Bitmaps	Schema zum Laden der Bitmap	Beschreibung
`ImageDecodingPolicy.ON_DEMAND`	Synchron	Geladene Bilder werden beim Zugriff auf die Bilddaten dekodiert. Verwenden Sie diese Richtlinie zum Dekodieren von kleineren Bildern. Diese Richtlinie eignet sich auch, wenn Ihre Anwendung keine komplexen Effekte und Übergänge verwendet.
`ImageDecodingPolicy.ON_LOAD`	Asynchron	Geladene Bilder werden beim Laden und vor dem Auslösen des `COMPLETE`-Ereignisses dekodiert. Diese Richtlinie eignet sich ideal für größere Bilder (größer als 10 MP). Wenn Sie AIR-Anwendungen für Mobilgeräte mit Seitenübergängen entwickeln, sollten Sie diese Richtlinie zum Laden von Bitmaps verwenden, um die wahrgenommene Leistung Ihrer Anwendung zu verbessern.

Hinweis: *Wenn es sich bei der geladenen Datei um ein Bitmapbild handelt und die Dekodierungsrichtlinie `ON_LOAD` verwendet wird, wird das Bild asynchron dekodiert, bevor das `COMPLETE`-Ereignis ausgelöst wird.*

Der folgende Code zeigt die Verwendung der `ImageDecodingPolicy`-Klasse:

```
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.imageDecodingPolicy = ImageDecodingPolicy.ON_LOAD
var loader:Loader = new Loader();
loader.load(new URLRequest("http://www.adobe.com/myimage.png"), loaderContext);
```

Sie können die `ON_DEMAND`-Kodierung weiterhin mit den Methoden `Loader.load()` und `Loader.loadBytes()` verwenden. Alle anderen Methoden, die ein `LoaderContext`-Objekt als Argument verwenden, ignorieren jedoch den für `ImageDecodingPolicy` übergebenen Wert.

Das folgende Beispiel veranschaulicht den Unterschied zwischen der synchronen und der asynchronen Dekodierung eines Bitmapbildes.

```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.system.ImageDecodingPolicy;
    import flash.system.LoaderContext;

    public class AsyncTest extends Sprite
    {
        private var loaderContext:LoaderContext;
        private var loader:Loader;
        private var urlRequest:URLRequest;
        public function AsyncTest()
        {
            //Load the image synchronously
            loaderContext = new LoaderContext();
            //Default behavior.
            loaderContext.imageDecodingPolicy = ImageDecodingPolicy.ON_DEMAND;
            loader = new Loader();
            loadImageSync();

            //Load the image asynchronously
            loaderContext = new LoaderContext();
            loaderContext.imageDecodingPolicy = ImageDecodingPolicy.ON_LOAD;
            loader = new Loader();
            loadImageASync();
        }

        private function loadImageASync():void{
            trace("Loading image asynchronously...");
            urlRequest = new URLRequest("http://www.adobe.com/myimage.png");
            urlRequest.useCache = false;
            loader.load(urlRequest, loaderContext);
            loader.contentLoaderInfo.addEventListener
                (Event.COMPLETE, onAsyncLoadComplete);
```

```
        }

        private function onAsyncLoadComplete(event:Event):void{
            trace("Async. Image Load Complete");
        }

        private function loadImageSync():void{
            trace("Loading image synchronously...");
            urlRequest = new URLRequest("http://www.adobe.com/myimage.png");
            urlRequest.useCache = false;
            loader.load(urlRequest, loaderContext);
            loader.contentLoaderInfo.addEventListener
                (Event.COMPLETE, onSyncLoadComplete);
        }

        private function onSyncLoadComplete(event:Event):void{
            trace("Sync. Image Load Complete");
        }
    }
}
```

Eine Demonstration der Auswirkung der verschiedenen Dekodierungsrichtlinien finden Sie unter Thibaud Imbert: Asynchronous bitmap decoding in the Adobe Flash runtimes.

Kapitel 14: Anwenden von Filtern auf Anzeigeobjekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Vergangenheit war die Anwendung von Filtereffekten auf Bitmapgrafiken die Domäne von speziellen Bildbearbeitungsprogrammen wie Adobe Photoshop® und Adobe Fireworks®. ActionScript 3.0 umfasst das flash.filters-Paket, das verschiedene Bitmap-Effektfilterklassen enthält. Mit diesen Effekten können Entwickler programmgesteuert Filter auf Bitmap- und Anzeigeobjekte anwenden, um viele der sonst nur in Bildbearbeitungsprogrammen verfügbaren Effekte zu erzielen.

Grundlagen der Anwendung von Filtern auf Anzeigeobjekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine Möglichkeit, eine Anwendung aufzuwerten, ist das Hinzufügen von einfachen Grafikeffekten. Sie können beispielsweise einen Schlagschatten hinter einem Foto hinzufügen, mit dem ein dreidimensionaler Eindruck erweckt wird, oder Glühen um eine Schaltfläche, das den aktivierten Status signalisiert. ActionScript 3.0 umfasst zehn Filter, die Sie auf beliebige Anzeigeobjekte und BitmapData-Instanzen anwenden können. Das Angebot reicht von einfachen Filtern wie Schlagschatten und Glühen bis hin zu komplexen Filtern, z. B. Verschiebungsmatrix- und Convolution-Filter.

Hinweis: *Zusätzlich zu den integrierten Filtern können Sie mithilfe von Pixel Bender auch benutzerdefinierte Filter und Effekte programmieren. Lesen Sie dazu „Arbeiten mit Pixel Bender-Shadern“ auf Seite 321.*

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe aufgeführt, die beim Erstellen von Filtern verwendet werden:

Geschliffen Durch das Aufhellen der Pixel an zwei Seite und Abdunkeln der Pixel an den gegenüberliegenden Seiten entsteht eine abgeflachte Kante. Dieser Effekt erweckt den Eindruck eines dreidimensionalen Randes. Er wird häufig für erhobene oder gedrückte Schaltflächen und ähnliche Grafiken eingesetzt.

Verzerren Das Verzerren von Pixeln in einem Bild durch das Kombinieren jedes Pixelwerts mit den Werten einiger oder aller benachbarter Pixel. Dabei werden verschiedene Berechnungsverhältnisse angewendet.

Verschiebung Das Verschieben oder Bewegen von Pixeln in einem Bild an eine neue Position.

Matrix Ein Zahlenraster, das zum Durchführen bestimmter mathematischer Berechnungen verwendet wird, indem die Zahlen im Raster auf verschiedene Werte angewendet und die Ergebnisse dann zusammengefasst werden.

Verwandte Hilfethemen

flash.filters-Paket

flash.display.DisplayObject.filters

flash.display.BitmapData.applyFilter()

Erstellen und Anwenden von Filtern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit Filtern können Sie verschiedene Effekte auf Bitmap- und Anzeigeobjekte anwenden, von Schlagschatten bis hin zu abgeschrägten Kanten und Weichzeichnungen. Jeder Filter ist als eine Klasse definiert, daher müssen beim Anwenden von Filtern Instanzen der Filterobjekte erstellt werden. Dies verhält sich genauso wie beim Erstellen anderer Objekte. Nachdem Sie eine Instanz eines Filterobjekts erstellt haben, können Sie es mithilfe der `filters`-Eigenschaft des Objekts (oder bei einem BitmapData-Objekt mithilfe der `applyFilter()`-Methode) auf ein Anzeigeobjekt anwenden.

Erstellen eines Filters

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Erstellen eines Filterobjekts rufen Sie einfach die Konstruktormethode der ausgewählten Filterklasse auf. Um beispielsweise ein DropShadowFilter-Objekt zu erstellen, verwenden Sie den folgenden Code:

```
import flash.filters.DropShadowFilter;
var myFilter:DropShadowFilter = new DropShadowFilter();
```

Der `DropShadowFilter()`-Konstruktor akzeptiert (wie alle Konstruktoren von Filterklassen) verschiedene optionale Parameter, mit denen das Erscheinungsbild des Filtereffekts angepasst werden kann. Dies wird hier jedoch nicht näher beschrieben.

Anwenden eines Filters

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Nachdem Sie ein Filterobjekt erstellt haben, können Sie es auf ein Anzeigeobjekt oder ein BitmapData-Objekt anwenden. Wie der Filter genau angewendet wird, hängt von dem Objekt ab, auf das Sie ihn anwenden möchten.

Anwenden eines Filters auf ein Anzeigeobjekt

Zum Anwenden von Filtereffekten auf ein Anzeigeobjekt können Sie die `filters`-Eigenschaft verwenden. Die `filters`-Eigenschaft eines Anzeigeobjekts ist eine Array-Instanz. Deren Elemente sind die Filterobjekte, die auf das Anzeigeobjekt angewendet werden. Um einen einzelnen Filter auf ein Anzeigeobjekt anzuwenden, erstellen Sie zunächst eine Filterinstanz. Fügen Sie sie dann einer Array-Instanz hinzu und weisen Sie dieses Array-Objekt der `filters`-Eigenschaft des Anzeigeobjekts zu:

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.filters.DropShadowFilter;

// Create a bitmapData object and render it to screen
var myBitmapData:BitmapData = new BitmapData(100,100,false,0xFFFF3300);
var myDisplayObject:Bitmap = new Bitmap(myBitmapData);
addChild(myDisplayObject);

// Create a DropShadowFilter instance.
var dropShadow:DropShadowFilter = new DropShadowFilter();

// Create the filters array, adding the filter to the array by passing it as
// a parameter to the Array() constructor.
var filtersArray:Array = new Array(dropShadow);

// Assign the filters array to the display object to apply the filter.
myDisplayObject.filters = filtersArray;
```

Wenn Sie mehrere Filter auf ein Objekt anwenden möchten, fügen Sie einfach alle Filter zur Array-Instanz hinzu, bevor Sie das Array-Objekt der `filters`-Eigenschaft zuweisen. Sie können einem Array mehrere Objekte hinzufügen, indem Sie sie als Parameter an den Array-Konstruktor übergeben. Mit dem folgenden Code wird beispielsweise ein Geschliffen- (bevel) und einen Glühen-Filter (glow) auf das im vorangegangenen Beispiel erstellte Anzeigeobjekt angewendet:

```
import flash.filters.BevelFilter;
import flash.filters.GlowFilter;

// Create the filters and add them to an array.
var bevel:BevelFilter = new BevelFilter();
var glow:GlowFilter = new GlowFilter();
var filtersArray:Array = new Array(bevel, glow);

// Assign the filters array to the display object to apply the filter.
myDisplayObject.filters = filtersArray;
```

Zum Erstellen des Arrays mit den Filtern können Sie entweder den `new Array()`-Konstruktor (wie in den vorangegangenen Beispielen) oder die Array-Literalsyntax verwenden, bei der die Filter in eckigen Klammern eingeschlossen sind (`[]`). Beispielsweise führt die folgende Codezeile:

```
var filters:Array = new Array(dropShadow, blur);
```

das Gleiche aus wie diese Codezeile:

```
var filters:Array = [dropShadow, blur];
```

Wenn Sie mehrere Filter auf Anzeigeobjekte anwenden, werden sie kumulativ und aufeinander folgend angewendet. Angenommen, ein filters-Array enthält zwei Elemente, zunächst einen Geschliffen-Filter und dann einen Schlagschatten-Filter, so wird erst der Geschliffen-Filter auf das Anzeigeobjekt angewendet und dann der Schlagschatten-Filter auf den Geschliffen-Filter und das Anzeigeobjekt. Der Grund hierfür ist, dass sich der Schlagschatten-Filter an zweiter Stelle im filters-Array befindet. Wenn Sie Filter nicht kumulativ anwenden möchten, wenden Sie jeden Filter auf eine neue Kopie des Anzeigeobjekts an.

Sollen einem Anzeigeobjekt nur ein oder wenige Filter zugewiesen werden, können Sie in nur einer Anweisung die Filterinstanz erstellen und dem Objekt zuweisen. Mit der folgenden Codezeile wird beispielsweise ein Weichzeichnen-Filter auf ein Anzeigeobjekt mit dem Namen `myDisplayObject` angewendet:

```
myDisplayObject.filters = [new BlurFilter()];
```

Mit diesem Code wird mit der Array-Literalsyntax (eckige Klammern) eine Array-Instanz erstellt. Anschließend wird eine BlurFilter-Instanz als Element dieses Arrays erstellt und das Array dann der `filters`-Eigenschaft des Anzeigeobjekts `myDisplayObject` zugewiesen.

Entfernen von Filtern von einem Anzeigeobjekt

Das Entfernen aller Filter von einem Anzeigeobjekt ist sehr einfach. Sie müssen lediglich der `filters`-Eigenschaft den Wert null zuweisen:

```
myDisplayObject.filters = null;
```

Wenn Sie mehrere Filter auf ein Objekt angewendet haben und nur einen Filter entfernen möchten, müssen Sie mehrere Schritte ausführen, um das Array der `filters`-Eigenschaft zu ändern. Weitere Informationen finden Sie unter „Potenzielle Probleme beim Verwenden von Filtern“ auf Seite 289.

Anwenden eines Filters auf ein BitmapData-Objekt

Zum Anwenden eines Filters auf ein BitmapData-Objekt ist die `applyFilter()`-Methode des BitmapData-Objekts erforderlich:

```
var rect:Rectangle = new Rectangle();
var origin:Point = new Point();
myBitmapData.applyFilter(sourceBitmapData, rect, origin, new BlurFilter());
```

Die `applyFilter()`-Methode wendet einen Filter auf das BitmapData-Quellobjekt an und erzeugt ein neues gefiltertes Bild. Bei dieser Methode wird das ursprüngliche Bild nicht geändert. Stattdessen wird das Ergebnis des auf das Quellbild angewendeten Filters in der BitmapData-Instanz gespeichert, für die die `applyFilter()`-Methode aufgerufen wird.

Funktionsweise von Filtern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Anwenden von Filtern auf Anzeigeobjekte wird eine Kopie des ursprünglichen Objekts als eine transparentes Bitmap zwischengespeichert.

Nachdem ein Filter auf ein Anzeigeobjekt angewendet wurde, wird das Objekt in der Laufzeitumgebung als Bitmap zwischengespeichert, solange es über eine gültige Filterliste verfügt. Diese Quellbitmap wird dann als Originalbild für alle nachfolgend angewendeten Filtereffekte verwendet.

Jedes Anzeigeobjekt besitzt in der Regel zwei Bitmaps: eine mit dem ursprünglichen Quellanzeigeobjekt und eine zweite für das nach dem Filtern entstehende Bild. Dieses Ergebnisbild wird für die Darstellung verwendet. Solange sich das Anzeigeobjekt nicht ändert, muss das Ergebnisbild nicht aktualisiert werden.

Potenzielle Probleme beim Verwenden von Filtern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Verwenden von Filtern können verschiedene potenzielle Probleme und Fehler auftreten.

Filter und Bitmap-Zwischenspeicherung

Um einen Filter auf ein Anzeigeobjekt anwenden zu können, muss die Bitmap-Zwischenspeicherung für das Objekt aktiviert sein. Wenn Sie einen Filter auf ein Anzeigeobjekt anwenden, dessen `cacheAsBitmap`-Eigenschaft auf `false` gesetzt ist, wird die `cacheAsBitmap`-Eigenschaft des Objekts automatisch auf `true` gesetzt. Wenn Sie später sämtliche Filter eines Anzeigeobjekts entfernen, wird die `cacheAsBitmap`-Eigenschaft auf den Wert zurückgesetzt, der zuletzt eingestellt war.

Ändern von Filtern zur Laufzeit

Wenn einem Anzeigeobjekt bereits ein oder mehrere Filter zugewiesen wurden, können Sie diesen Filtersatz nicht ändern, indem Sie zusätzliche Filter hinzufügen oder Filter aus dem Array der `filters`-Eigenschaft entfernen. Um den bereits angewendeten Filtersatz zu erweitern oder zu ändern, müssen Sie die Änderungen an einem separaten Array vornehmen und dieses Array dann der filters-Eigenschaft des Anzeigeobjekts für die auf das Objekt anzuwendenden Filter zuweisen. Die einfachste Methode dafür ist das Einlesen des Arrays der `filters`-Eigenschaft in eine Array-Variable und das Vornehmen von Änderungen in diesem temporären Array. Anschließend weisen Sie das Array wieder der `filters`-Eigenschaft des Anzeigeobjekts zu. In komplexeren Fällen müssen Sie unter Umständen ein separates Master-Filter-Array verwenden. Sie nehmen dann alle Änderungen an diesem Master-Filter-Array vor und weisen das Master-Array der `filters`-Eigenschaft des Anzeigeobjekts nach jeder Änderung erneut zu.

Hinzufügen eines zusätzlichen Filters

Der folgende Code zeigt das Hinzufügen eines zusätzlichen Filters zu einem Anzeigeobjekt, das schon einen oder mehrere Filter aufweist. Zunächst wurde ein Glühen-Filter auf ein Anzeigeobjekt namens `myDisplayObject` angewendet. Beim Klicken auf das Anzeigeobjekt wird dann die Funktion `addFilters()` aufgerufen. In dieser Funktion werden zwei zusätzliche Filter auf `myDisplayObject` angewendet:

```
import flash.events.MouseEvent;
import flash.filters.*;

myDisplayObject.filters = [new GlowFilter()];

function addFilters(event:MouseEvent):void
{
    // Make a copy of the filters array.
    var filtersCopy:Array = myDisplayObject.filters;

    // Make desired changes to the filters (in this case, adding filters).
    filtersCopy.push(new BlurFilter());
    filtersCopy.push(new DropShadowFilter());

    // Apply the changes by reassigning the array to the filters property.
    myDisplayObject.filters = filtersCopy;
}

myDisplayObject.addEventListener(MouseEvent.CLICK, addFilters);
```

Entfernen eines Filters aus einem Filtersatz

Wenn mehrere Filter auf ein Anzeigeobjekt angewendet wurden und Sie einen der Filter entfernen möchten, während die anderen Filter weiter auf das Objekt angewendet sind, können Sie die Filter in ein temporäres Array kopieren, den unerwünschten Filter aus diesem Array entfernen und das temporäre Array der `filters`-Eigenschaft des Anzeigeobjekts neu zuweisen. Verschiedene Möglichkeiten zum Entfernen eines oder mehrerer Elemente aus einem Array werden im Abschnitt „Abrufen von Werten und Entfernen von Array-Elementen“ auf Seite 32 beschrieben.

Die einfachste Möglichkeit ist das Entfernen des obersten Filters für das Objekt (der zuletzt auf das Objekt angewendete Filter). Sie verwenden die `pop()`-Methode der Array-Klasse zum Entfernen des Filters aus dem Array:

```
// Example of removing the top-most filter from a display object
// named "filteredObject".

var tempFilters:Array = filteredObject.filters;

// Remove the last element from the Array (the top-most filter).
tempFilters.pop();

// Apply the new set of filters to the display object.
filteredObject.filters = tempFilters;
```

Zum Entfernen des untersten Filters (des zuerst auf das Objekt angewendeten Filters) verwenden Sie den gleichen Code, wobei Sie jedoch die `shift()`-Methode der Array-Klasse anstatt der `pop()`-Methode verwenden.

Zum Entfernen eines Filters aus der Mitte eines Filter-Arrays (wenn das Array mehr als zwei Filter hat) können Sie die `splice()`-Methode verwenden. Sie müssen die Indexposition (die Position im Array) des Filters kennen, den Sie entfernen möchten. Mit dem folgenden Code wird z. B. der zweite Filter (Filter mit Indexposition 1) von einem Anzeigeobjekt entfernt:

```
// Example of removing a filter from the middle of a stack of filters
// applied to a display object named "filteredObject".

var tempFilters:Array = filteredObject.filters;

// Remove the second filter from the array. It's the item at index 1
// because Array indexes start from 0.
// The first "1" indicates the index of the filter to remove; the
// second "1" indicates how many elements to remove.
tempFilters.splice(1, 1);

// Apply the new set of filters to the display object.
filteredObject.filters = tempFilters;
```

Ermitteln der Indexposition eines Filters

Sie müssen wissen, welcher Filter aus dem Array entfernt werden soll, um die Indexposition des Filters zu kennen. Sie müssen die Indexposition des zu entfernenden Filters entweder kennen (anhand des Aufbaus der Anwendung) oder berechnen.

Der beste Ansatz ist dabei, die Anwendung so zu konzipieren, dass sich der zu entfernende Filter immer an der gleichen Position innerhalb des Filtersatzes befindet. Wenn z. B. ein einziges Anzeigeobjekt vorliegt, auf das zuerst ein Convolution-Filter und dann ein Schlagschatten-Filter angewendet wurde, und Sie den Schlagschatten-Filter entfernen, den Convolution-Filter aber beibehalten möchten, wissen Sie, an welcher Position sich der zu entfernende Filter befindet (oberste Position), sodass die zu verwendende Array-Methode leicht zu ermitteln ist (in diesem Beispiel `Array.pop()` zum Entfernen des Schlagschatten-Filters).

Wenn der zu entfernende Filter immer einen bestimmten Typ aufweist, aber sich nicht immer unbedingt an der gleichen Position innerhalb des Filtersatzes befindet, können Sie die Datentypen der einzelnen Filter im Array prüfen, um zu ermitteln, welcher Filter entfernt werden soll. Mit dem folgenden Code wird z. B. ermittelt, welcher Filter innerhalb eines Filtersatzes ein Glühen-Filter ist, und dieser Filter aus dem Satz entfernt.

```
// Example of removing a glow filter from a set of filters, where the
//filter you want to remove is the only GlowFilter instance applied
// to the filtered object.

var tempFilters:Array = filteredObject.filters;

// Loop through the filters to find the index of the GlowFilter instance.
var glowIndex:int;
var numFilters:int = tempFilters.length;
for (var i:int = 0; i < numFilters; i++)
{
    if (tempFilters[i] is GlowFilter)
    {
        glowIndex = i;
        break;
    }
}

// Remove the glow filter from the array.
tempFilters.splice(glowIndex, 1);

// Apply the new set of filters to the display object.
filteredObject.filters = tempFilters;
```

In komplexeren Fällen, wenn der zu entfernende Filter z. B. zur Laufzeit ausgewählt wird, besteht der beste Ansatz in der Erstellung einer separaten, dauerhaften Kopie des Filter-Arrays, die als Master-Filterliste fungiert. Jedes Mal, wenn Sie eine Änderung am Filtersatz vornehmen, ändern Sie die Master-Liste und wenden Sie dann dieses Filter-Array als `filters`-Eigenschaft des Anzeigeobjekts an.

Im folgenden Codebeispiel werden z. B. mehrere Convolution-Filter auf ein Anzeigeobjekt angewendet, um verschiedene visuelle Effekte zu erzeugen. Später wird einer dieser Filter entfernt, während die anderen beibehalten werden. In diesem Beispiel werden eine Master-Kopie des Filter-Arrays sowie ein Verweis auf den zu entfernenden Filter erstellt. Das Auffinden und Entfernen des speziellen Filters ähnelt dem vorhergehenden Verfahren, nur wird keine temporäre Kopie des Filter-Arrays erstellt, sondern die Master-Kopie wird bearbeitet und auf das Anzeigeobjekt angewendet.

```
// Example of removing a filter from a set of
// filters, where there may be more than one
// of that type of filter applied to the filtered
// object, and you only want to remove one.

// A master list of filters is stored in a separate,
// persistent Array variable.
var masterFilterList:Array;

// At some point, you store a reference to the filter you
// want to remove.
var filterToRemove:ConvolutionFilter;

// ... assume the filters have been added to masterFilterList,
// which is then assigned as the filteredObject.filters:
filteredObject.filters = masterFilterList;

// ... later, when it's time to remove the filter, this code gets called:

// Loop through the filters to find the index of masterFilterList.
var removeIndex:int = -1;
var numFilters:int = masterFilterList.length;
for (var i:int = 0; i < numFilters; i++)
{
    if (masterFilterList[i] == filterToRemove)
    {
        removeIndex = i;
        break;
    }
}

if (removeIndex >= 0)
{
    // Remove the filter from the array.
    masterFilterList.splice(removeIndex, 1);

    // Apply the new set of filters to the display object.
    filteredObject.filters = masterFilterList;
}
```

Bei diesem Verfahren (Vergleichen eines gespeicherten Filterverweises mit den Elementen im Filter-Array, um den zu entfernenden Filter zu ermitteln) *müssen* Sie eine separate Kopie des Filter-Arrays erstellen – der Code funktioniert nicht, wenn Sie den gespeicherten Filterverweis mit den Elementen in einem temporären Array vergleichen, das von der `filters`-Eigenschaft des Anzeigeobjekts kopiert wurde. Dies liegt daran, dass die Laufzeitumgebung Kopien der Filterobjekte im Array erstellt, wenn Sie ein Array zur `filters`-Eigenschaft zuweisen. Diese Kopien (und nicht die Originalobjekte) werden auf das Anzeigeobjekt angewendet; wenn Sie die `filters`-Eigenschaft in ein temporäres Array einlesen, enthält das temporäre Array Verweise auf die kopierten Filterobjekte, und nicht auf die Original-Filterobjekte. Wenn Sie dementsprechend im vorangehenden Beispiel versuchen, die Indexposition von `filterToRemove` zu ermitteln, indem Sie ihn mit den Filtern in einem temporären Filter-Array vergleichen, wird keine Übereinstimmung gefunden.

Filter und Objekttransformationen

Gefilterte Regionen (z. B. ein Schlagschatten) außerhalb des Begrenzungsrahmens eines Anzeigeobjekts werden bei der Kollisionserkennung (beim Ermitteln, ob eine Instanz eine andere Instanz überlappt oder schneidet) nicht als Teil der Oberfläche berücksichtigt. Da die Kollisionserkennung der DisplayObject-Klasse vektorbasiert ist, können Sie keine Kollisionserkennung am Bitmapergebnis vornehmen. Wenn Sie z. B. einen Geschliffen-Filter auf eine Schaltflächeninstanz anwenden, steht die Kollisionserkennung für den entsprechenden Teil der Instanz nicht zur Verfügung.

Skalierung, Drehung und Neigung werden nicht von Filtern unterstützt. Zwar ist das gefilterte Anzeigeobjekt selbst skaliert (wenn `scaleX` und `scaleY` nicht 100 % sind), der Filtereffekt wird jedoch nicht zusammen mit der Instanz skaliert. Das bedeutet, dass die ursprüngliche Form der Instanz gedreht, skaliert oder geneigt ist, der Filter diese Effekte jedoch nicht nachvollzieht.

Um realistische Effekte zu erstellen, können Sie eine Instanz mit einem Filter animieren, oder Sie erzielen den gleichen Effekt durch das Verschachteln von Instanzen und Animieren von Filtern mit der BitmapData-Klasse.

Filter und Bitmapobjekte

Wenn Sie einen Filter auf ein BitmapData-Objekt anwenden, wird die `cacheAsBitmap`-Eigenschaft automatisch auf `true` gesetzt. Auf diese Weise wird der Filter auf die Kopie des Objekts und nicht auf das ursprüngliche Objekt angewendet.

Diese Kopie wird dann auf der Hauptanzeige (über dem ursprünglichen Objekt) möglichst nah am nächsten Pixel platziert. Ändern sich die Begrenzungen der ursprünglichen Bitmap, wird die gefilterte Kopie der Bitmap nicht gestreckt oder verzerrt, sondern neu erstellt.

Wenn Sie alle Filter eines Anzeigeobjekts löschen, wird die `cacheAsBitmap`-Eigenschaft auf den Wert zurückgesetzt, der vor dem Anwenden der Filter gültig war.

Verfügbare Anzeigefilter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 umfasst zehn Filterklassen, die Sie auf Anzeigeobjekte und BitmapData-Objekte anwenden können:

- Geschliffen-Filter (BevelFilter-Klasse)
- Weichzeichnen-Filter (BlurFilter-Klasse)
- Schlagschatten-Filter (DropShadowFilter-Klasse)
- Glühen-Filter (GlowFilter-Klasse)
- Farbverlauf-Geschliffen-Filter (GradientBevelFilter-Klasse)
- Farbverlauf-Glühen-Filter (GradientGlowFilter-Klasse)
- Farbmatrix-Filter (ColorMatrixFilter-Klasse)
- Convolution-Filter (ConvolutionFilter-Klasse)
- Verschiebungsmatrix-Filter (DisplacementMapFilter-Klasse)
- Shader-Filter (ShaderFilter-Klasse)

Die ersten sechs Filter sind einfache Filter, mit denen ein bestimmter Effekt erzeugt wird. Dieser Effekt kann bis zu einem gewissen Grad angepasst werden. Diese sechs Filter können mithilfe von ActionScript angewendet werden. Außerdem können sie in Flash Professional über das Bedienfeld „Filter“ auf Objekte angewendet werden. Entsprechend können Sie, wenn Sie über Flash Professional verfügen, verschiedene Filter und Einstellungen schnell mit der visuellen Schnittstelle ausprobieren, um herauszufinden, wie Sie einen gewünschten Effekt erstellen. Dies gilt auch dann, wenn Sie die Filter mit ActionScript anwenden.

Die letzten vier Filter stehen nur in ActionScript zur Verfügung. Diese Filter – Farbmatrix-Filter, Convolution-Filter, Verschiebungsmatrix-Filter und Shader-Filter – sind wesentlich flexibler, was die Arten der mit ihnen erstellbaren Effekte angeht. Sie sind nicht für einen bestimmten Effekte optimiert, sondern bieten Leistungsstärke und Flexibilität. Indem Sie beispielsweise verschiedene Werte für die Matrix des Convolution-Filters auswählen, kann dieser zum Erstellen von Effekten wie Weichzeichnen, Prägen, Schärfen, Suchen von Farbrändern, Transformationen und mehr verwendet werden.

Jeder Filter, ob einfach oder komplex, kann über dessen Eigenschaften angepasst werden. Im Allgemeinen stehen Ihnen zwei Möglichkeiten beim Einstellen der Filtereigenschaften zur Verfügung. Bei allen Filtern können Sie die Eigenschaften durch Übergabe von Parameterwerten an den Konstruktor des Filterobjekts einstellen. Alternativ können Sie die Filter durch Einstellen der Werte der Filterobjekteigenschaften anpassen. Dies ist unabhängig davon möglich, ob Sie die Filtereigenschaften durch Übergabe von Parametern einstellen. In den Codebeispielen werden die Eigenschaften im Wesentlichen direkt eingestellt, damit das Beispiel leichter lesbar ist. Trotzdem können Sie das gleiche Ergebnis in der Regel mit weniger Codezeilen erreichen, indem Sie die Werte als Parameter im Konstruktor des Filterobjekts übergeben. Ausführliche Informationen zu den einzelnen Filtern sowie ihren Eigenschaften und Konstruktorparametern finden Sie im Eintrag zum flash.filters-Paket im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Geschliffen-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der BevelFilter-Klasse können Sie dem gefilterten Objekt eine geschliffen wirkenden 3D-Kante hinzufügen. Dieser Filter lässt die scharfen Kanten oder Ränder Ihres Objekts so erscheinen, als ob sie mit einem Meißel bearbeitet oder abgeschrägt worden wären.

Mit den Eigenschaften der BevelFilter-Klasse können Sie das Erscheinungsbild der abgeschrägten Kante anpassen. Sie können Farben zum Hervorheben und Schattieren einstellen, Weichzeichnungen, Winkel und Platzierungen abschrägen und sogar einen Aussparungseffekt schaffen.

Im folgenden Beispiel wird ein externes Bild geladen und ein Geschliffen-Filter auf das Bild angewendet.

```
import flash.display.*;
import flash.filters.BevelFilter;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.net.URLRequest;

// Load an image onto the Stage.
var imageLoader:Loader = new Loader();
var url:String = "http://www.helpexamples.com/flash/images/image3.jpg";
var urlReq:URLRequest = new URLRequest(url);
imageLoader.load(urlReq);
addChild(imageLoader);

// Create the bevel filter and set filter properties.
var bevel:BevelFilter = new BevelFilter();

bevel.distance = 5;
bevel.angle = 45;
bevel.highlightColor = 0xFFFF00;
bevel.highlightAlpha = 0.8;
bevel.shadowColor = 0x666666;
bevel.shadowAlpha = 0.8;
bevel.blurX = 5;
bevel.blurY = 5;
bevel.strength = 5;
bevel.quality = BitmapFilterQuality.HIGH;
bevel.type = BitmapFilterType.INNER;
bevel.knockout = false;

// Apply filter to the image.
imageLoader.filters = [bevel];
```

Weichzeichnen-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der BlurFilter-Klasse wird ein Anzeigeobjekt mit seinem Inhalt verwischt oder weichgezeichnet dargestellt. Weichzeichnen-Effekte eignen sich besonders dann, wenn der Eindruck erweckt werden soll, dass sich ein Objekt außerhalb des Fokus befindet, oder wenn schnelle Bewegungen simuliert werden sollen. Mit einem niedrigen Wert für die `quality`-Eigenschaft wird ein leicht verschwommener Effekt simuliert. Mit einem hohen Wert für die `quality`-Eigenschaft wird ein sanftes Weichzeichnen erzeugt, das dem Effekt Gauß-Verwischen ähnelt.

Im folgenden Beispiel wird ein Kreisobjekt mit der `drawCircle()`-Methode der Graphics-Klasse erstellt und ein Weichzeichnen-Filter auf das Objekt angewendet:

```
import flash.display.Sprite;
import flash.filters.BitmapFilterQuality;
import flash.filters.BlurFilter;

// Draw a circle.
var redDotCutout:Sprite = new Sprite();
redDotCutout.graphics.lineStyle();
redDotCutout.graphics.beginFill(0xFF0000);
redDotCutout.graphics.drawCircle(145, 90, 25);
redDotCutout.graphics.endFill();

// Add the circle to the display list.
addChild(redDotCutout);

// Apply the blur filter to the rectangle.
var blur:BlurFilter = new BlurFilter();
blur.blurX = 10;
blur.blurY = 10;
blur.quality = BitmapFilterQuality.MEDIUM;
redDotCutout.filters = [blur];
```

Schlagschatten-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Schlagschatten erwecken den Eindruck, als ob sich eine separate Lichtquelle über dem Zielobjekt befindet. Position und Intensität dieser Lichtquelle können geändert werden, um verschiedene Schlagschatteneffekte zu erzeugen.

Die DropShadowFilter-Klasse verwendet einen Algorithmus, der dem des Weichzeichnen-Filters ähnelt. Der wesentliche Unterschied besteht darin, dass der Schlagschatten-Filter über mehr Eigenschaften verfügt, die zum Simulieren von verschiedenen Lichtquellen-Attributen (zum Beispiel Alpha, Farbe, Versatz und Helligkeit) eingestellt werden können.

Darüber hinaus können Sie mit dem Schlagschatten-Filter benutzerdefinierte Transformationsoptionen auf den Stil eines Schlagschattens anwenden, z. B. einen Innen- oder Außenschatten und einen Aussparungsmodus.

Mit dem folgenden Code wird ein quadratisches Sprite-Objekt erstellt und ein Schlagschatten-Filter auf das Objekt angewendet:

```
import flash.display.Sprite;
import flash.filters.DropShadowFilter;

// Draw a box.
var boxShadow:Sprite = new Sprite();
boxShadow.graphics.lineStyle(1);
boxShadow.graphics.beginFill(0xFF3300);
boxShadow.graphics.drawRect(0, 0, 100, 100);
boxShadow.graphics.endFill();
addChild(boxShadow);

// Apply the drop shadow filter to the box.
var shadow:DropShadowFilter = new DropShadowFilter();
shadow.distance = 10;
shadow.angle = 25;

// You can also set other properties, such as the shadow color,
// alpha, amount of blur, strength, quality, and options for
// inner shadows and knockout effects.

boxShadow.filters = [shadow];
```

Glühen-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die GlowFilter-Klasse wendet einen Beleuchtungseffekt auf Anzeigeobjekte an. Dadurch entsteht der Eindruck, dass ein Licht von unten auf das Objekt gerichtet wird, wodurch ein weiches Glühen entsteht.

Ähnlich dem Schlagschatten-Filter verfügt der Glühen-Filter über Eigenschaften, über die Entfernung, Winkel und Farbe der Lichtquelle geändert werden können, sodass verschiedene Effekte möglich werden. Darüber hinaus verfügt der Glühen-Filter über verschiedene Optionen zum Ändern der Art des Glühens, z. B. einem Innen- oder Außenglühen und einem Aussparungsmodus.

Mit dem folgenden Code wird ein Kreuz mithilfe der Sprite-Klasse erstellt und ein Glühen-Filter auf dieses Objekt angewendet:

```
import flash.display.Sprite;
import flash.filters.BitmapFilterQuality;
import flash.filters.GlowFilter;

// Create a cross graphic.
var crossGraphic:Sprite = new Sprite();
crossGraphic.graphics.lineStyle();
crossGraphic.graphics.beginFill(0xCCCC00);
crossGraphic.graphics.drawRect(60, 90, 100, 20);
crossGraphic.graphics.drawRect(100, 50, 20, 100);
crossGraphic.graphics.endFill();
addChild(crossGraphic);

// Apply the glow filter to the cross shape.
var glow:GlowFilter = new GlowFilter();
glow.color = 0x009922;
glow.alpha = 1;
glow.blurX = 25;
glow.blurY = 25;
glow.quality = BitmapFilterQuality.MEDIUM;

crossGraphic.filters = [glow];
```

Farbverlauf-Geschliffen-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der GradientBevelFilter-Klasse können Sie einen erweiterten Geschliffen-Effekt auf Anzeigeobjekte oder BitmapData-Objekte anwenden. Das Anwenden eines Farbverlaufs auf den abgeschrägten Bereich verbessert die Tiefenwirkung des Geschliffen-Effekts und verleiht den Kanten einen realistischeren 3D-Effekt.

Im folgenden Code wird mit der `drawRect()`-Methode der Shape-Klasse ein rechteckiges Objekt erstellt und ein Farbverlauf-Geschliffen-Filter auf das Objekt angewendet.

```
import flash.display.Shape;
import flash.filters.BitmapFilterQuality;
import flash.filters.GradientBevelFilter;

// Draw a rectangle.
var box:Shape = new Shape();
box.graphics.lineStyle();
box.graphics.beginFill(0xFEFE78);
box.graphics.drawRect(100, 50, 90, 200);
box.graphics.endFill();

// Apply a gradient bevel to the rectangle.
var gradientBevel:GradientBevelFilter = new GradientBevelFilter();

gradientBevel.distance = 8;
gradientBevel.angle = 225; // opposite of 45 degrees
gradientBevel.colors = [0xFFFFCC, 0xFEFE78, 0x8F8E01];
gradientBevel.alphas = [1, 0, 1];
gradientBevel.ratios = [0, 128, 255];
gradientBevel.blurX = 8;
gradientBevel.blurY = 8;
gradientBevel.quality = BitmapFilterQuality.HIGH;

// Other properties let you set the filter strength and set options
// for inner bevel and knockout effects.

box.filters = [gradientBevel];

// Add the graphic to the display list.
addChild(box);
```

Farbverlauf-Glühen-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der GradientGlowFilter-Klasse können Sie einen erweiterten Glühen-Effekt auf Anzeigeobjekte oder BitmapData-Objekte anwenden. Mit diesem Effekt haben Sie eine bessere Kontrolle über die Farben des Glühens und erzeugen einen realistischeren Glühen-Effekt. Darüber hinaus können Sie mit dem Farbverlauf-Glühen-Filter ein Glühen mit Farbverlauf auf die inneren, äußeren oder oberen Kanten eines Objekts anwenden.

Im folgenden Beispiel wird ein Kreis auf der Bühne erstellt und ein Farbverlauf-Glühen-Filter auf das Objekt angewendet. Wenn Sie die Maus weiter nach rechts und unten ziehen, erhöht sich das Ausmaß der Weichzeichnung in horizontaler bzw. vertikaler Richtung. Darüber hinaus wird jedes Mal, wenn Sie auf die Bühne klicken, die Stärke der Weichzeichnung erhöht.

```
import flash.events.MouseEvent;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.filters.GradientGlowFilter;

// Create a new Shape instance.
var shape:Shape = new Shape();

// Draw the shape.
shape.graphics.beginFill(0xFF0000, 100);
shape.graphics.moveTo(0, 0);
shape.graphics.lineTo(100, 0);
shape.graphics.lineTo(100, 100);
shape.graphics.lineTo(0, 100);
shape.graphics.lineTo(0, 0);
shape.graphics.endFill();

// Position the shape on the Stage.
addChild(shape);
shape.x = 100;
shape.y = 100;

// Define a gradient glow.
var gradientGlow:GradientGlowFilter = new GradientGlowFilter();
gradientGlow.distance = 0;
gradientGlow.angle = 45;
gradientGlow.colors = [0x000000, 0xFF0000];
gradientGlow.alphas = [0, 1];
gradientGlow.ratios = [0, 255];
gradientGlow.blurX = 10;
gradientGlow.blurY = 10;
gradientGlow.strength = 2;
gradientGlow.quality = BitmapFilterQuality.HIGH;
gradientGlow.type = BitmapFilterType.OUTER;

// Define functions to listen for two events.
function onClick(event:MouseEvent):void
{
    gradientGlow.strength++;
    shape.filters = [gradientGlow];
}

function onMouseMove(event:MouseEvent):void
{
    gradientGlow.blurX = (stage.mouseX / stage.stageWidth) * 255;
    gradientGlow.blurY = (stage.mouseY / stage.stageHeight) * 255;
    shape.filters = [gradientGlow];
}
stage.addEventListener(MouseEvent.CLICK, onClick);
stage.addEventListener(MouseEvent.MOUSE_MOVE, onMouseMove);
```

Beispiel: Kombinieren einfacher Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Im folgenden Codebeispiel werden mehrere einfache Filter verwendet und mit einem Timer kombiniert. Auf diese Weise werden sich wiederholende Aktionen erstellt, die eine animierte Verkehrsampel darstellen.

```
import flash.display.Shape;
import flash.events.TimerEvent;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
import flash.filters.GradientBevelFilter;
import flash.utils.Timer;

var count:Number = 1;
var distance:Number = 8;
var angleInDegrees:Number = 225; // opposite of 45 degrees
var colors:Array = [0xFFFFCC, 0xFEFE78, 0x8F8E01];
var alphas:Array = [1, 0, 1];
var ratios:Array = [0, 128, 255];
var blurX:Number = 8;
var blurY:Number = 8;
var strength:Number = 1;
var quality:Number = BitmapFilterQuality.HIGH;
var type:String = BitmapFilterType.INNER;
var knockout:Boolean = false;

// Draw the rectangle background for the traffic light.
var box:Shape = new Shape();
box.graphics.lineStyle();
box.graphics.beginFill(0xFEFE78);
box.graphics.drawRect(100, 50, 90, 200);
box.graphics.endFill();

// Draw the 3 circles for the three lights.
var stopLight:Shape = new Shape();
stopLight.graphics.lineStyle();
stopLight.graphics.beginFill(0xFF0000);
stopLight.graphics.drawCircle(145,90,25);
stopLight.graphics.endFill();

var cautionLight:Shape = new Shape();
cautionLight.graphics.lineStyle();
cautionLight.graphics.beginFill(0xFF9900);
cautionLight.graphics.drawCircle(145,150,25);
cautionLight.graphics.endFill();

var goLight:Shape = new Shape();
goLight.graphics.lineStyle();
goLight.graphics.beginFill(0x00CC00);
goLight.graphics.drawCircle(145,210,25);
goLight.graphics.endFill();

// Add the graphics to the display list.
addChild(box);
```

```
addChild(stopLight);
addChild(cautionLight);
addChild(goLight);

// Apply a gradient bevel to the traffic light rectangle.
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(distance, angleInDegrees,
colors, alphas, ratios, blurX, blurY, strength, quality, type, knockout);
box.filters = [gradientBevel];

// Create the inner shadow (for lights when off) and glow
// (for lights when on).
var innerShadow:DropShadowFilter = new DropShadowFilter(5, 45, 0, 0.5, 3, 3, 1, 1, true,
false);
var redGlow:GlowFilter = new GlowFilter(0xFF0000, 1, 30, 30, 1, 1, false, false);
var yellowGlow:GlowFilter = new GlowFilter(0xFF9900, 1, 30, 30, 1, 1, false, false);
var greenGlow:GlowFilter = new GlowFilter(0x00CC00, 1, 30, 30, 1, 1, false, false);

// Set the starting state of the lights (green on, red/yellow off).
stopLight.filters = [innerShadow];
cautionLight.filters = [innerShadow];
goLight.filters = [greenGlow];

// Swap the filters based on the count value.
function trafficControl(event:TimerEvent):void
{
    if (count == 4)
    {
        count = 1;
    }

    switch (count)
    {
        case 1:
            stopLight.filters = [innerShadow];
            cautionLight.filters = [yellowGlow];
            goLight.filters = [innerShadow];
            break;
        case 2:
            stopLight.filters = [redGlow];
            cautionLight.filters = [innerShadow];
            goLight.filters = [innerShadow];
            break;
        case 3:
            stopLight.filters = [innerShadow];
            cautionLight.filters = [innerShadow];
            goLight.filters = [greenGlow];
            break;
    }

    count++;
}

// Create a timer to swap the filters at a 3 second interval.
var timer:Timer = new Timer(3000, 9);
timer.addEventListener(TimerEvent.TIMER, trafficControl);
timer.start();
```

Farbmatrix-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die ColorMatrixFilter-Klasse dient zum Ändern der Farb- und Alphawerte des gefilterten Objekts. Dies ermöglicht Ihnen Änderungen der Sättigung, Farbtonrotation (Verschieben einer Palette von einem Farbbereich zu einem anderen), Änderungen bei den Luminanz-Alpha-Werten sowie andere Farbänderungseffekte, bei denen Werte eines Farbkanals und potenziell auf andere Farbkanäle angewendet werden.

Im Prinzip durchläuft der Filter nacheinander die Pixel im Quellbild und trennt jedes Pixel in dessen Rot-, Grün-, Blau- und Alpha-Komponenten. Dann multipliziert der Filter jeden dieser Werte mit in einer Farbmatrix bereitgestellten Werten und addiert die Ergebnisse, um den resultierenden Farbwert zu berechnen, der für dieses Pixel auf dem Bildschirm angezeigt wird. Die `matrix`-Eigenschaft des Filters ist ein Array von 20 Ziffern, die zur Berechnung der Ergebnisfarbe verwendet werden. Einzelheiten zu dem Algorithmus, mit dem die Farbwerte berechnet werden, finden Sie im Eintrag zur `matrix`-Eigenschaft der ColorMatrixFilter-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Convolution-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der ConvolutionFilter-Klasse können Sie zahlreiche Bildtransformationen auf Anzeigeobjekte oder BitmapData-Objekte anwenden, wie beispielsweise Weichzeichnen, Kantenerkennung, Scharfstellen, Prägen und Schliff.

Im Prinzip durchläuft in der Convolution-Filter nacheinander jedes Pixel im Quellbild und berechnet die Ergebnisfarbe des Pixels aus dem Wert des Pixels und seiner umgebenden Pixel. Eine Matrix, die ein Array aus numerischen Werten darstellt, gibt an, in welchem Umfang sich der Wert eines bestimmten benachbarten Pixels auf den Ergebniswert auswirkt.

Betrachten Sie den am häufigsten verwendeten Matrixtyp, eine 3 x 3-Matrix. Diese Matrix enthält neun Werte:

```
N   N   N
N   P   N
N   N   N
```

Wenn der Convolution-Filter auf ein bestimmtes Pixel angewendet wird, werden der Farbwert des Pixels selbst (in diesem Beispiel „P“) sowie die Werte der umgebenden Pixel (in diesem Beispiel „N“) untersucht. Durch Einstellen der Werte in der Matrix können Sie jedoch festlegen, welche Priorität bestimmte Pixel bei den Auswirkungen auf das Ergebnisbild haben.

Beispielsweise lässt die folgende Matrix, wenn sie mit einem Convolution-Filter angewendet wird, das Bild vollkommen unverändert:

```
0   0   0
0   1   0
0   0   0
```

Das Bild bleibt unverändert, da der ursprüngliche Pixelwert bei der Berechnung der Ergebnispixelfarbe die relative Stärke 1 aufweist, während die Werte der umgebenden Pixel die relative Stärke 0 aufweisen, d. h., ihre Farben wirken sich nicht auf das Ergebnisbild aus.

Entsprechend führt die folgende Matrix dazu, dass die Pixel des Bilds um ein Pixel nach links verschoben werden:

```
0   0   0
0   0   1
0   0   0
```

In diesem Fall ist zu beachten, dass das Pixel selbst keine Auswirkung auf den Wert des an dieser Stelle im Ergebnisbild angezeigten Pixels hat. Nur der Wert des Pixels auf der rechten Seite wird zur Berechnung des Ergebnispixelwerts verwendet.

In ActionScript erstellen Sie die Matrix als eine Kombination einer Array-Instanz mit den Werten und zweier Eigenschaften, welche die Anzahl an Zeilen und Spalten in der Matrix angeben. Im folgenden Beispiel wird ein Bild geladen, und wenn das Bild vollständig geladen ist, ein Convolution-Filter unter Verwendung der Matrix aus dem vorangegangenen Code auf das Bild angewendet:

```
// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

function applyFilter(event:MouseEvent):void
{
    // Create the convolution matrix.
    var matrix:Array = [0, 0, 0,
                                    0, 0, 1,
                                    0, 0, 0];

    var convolution:ConvolutionFilter = new ConvolutionFilter();
    convolution.matrixX = 3;
    convolution.matrixY = 3;
    convolution.matrix = matrix;
    convolution.divisor = 1;

    loader.filters = [convolution];
}

loader.addEventListener(MouseEvent.CLICK, applyFilter);
```

Die Auswirkung anderer Werte als 1 und 0 in der Matrix ist aus diesem Codebeispiel nicht ersichtlich. Beispielsweise führt die gleiche Matrix mit der Zahl 8 anstelle der 1 an der rechten Position die gleiche Aktion durch (Verschieben der Pixel nach links). Darüber hinaus wirkt sie sich auf die Farben des Bilds aus und macht sie achtmal heller. Dies liegt daran, dass die Ergebnispixelwerte durch Multiplizieren der Matrixwerte mit den ursprünglichen Pixelfarben, Addieren der Werte und Dividieren des Ergebnisses durch den Wert der `divisor`-Eigenschaft des Filters berechnet werden. Beachten Sie bei dem folgenden Beispielcode, dass die `divisor`-Eigenschaft auf 1 gesetzt ist. Allgemein gilt: Wenn die Helligkeit der Farben in etwa der im ursprünglichen Bild entsprechen soll, müssen Sie sicherstellen, dass der Divisor der Summe der Matrixwerte entspricht. Wenn die Werte in der Matrix 8 ergeben und der Divisor 1 lautet, ist das Ergebnisbild etwa achtmal heller als das ursprüngliche Bild.

Obwohl der Effekt dieser Matrix kaum erkennbar ist, können andere Matrixwerte zum Erstellen verschiedener Effekte verwendet werden. Im Folgenden sind verschiedene Standardsätze mit Matrixwerten für verschiedene Effekte aufgeführt, die mithilfe einer 3 x 3-Matrix erstellt werden:

- Einfache Weichzeichnung (Divisor 5):

```
 0 1 0
 1 1 1
 0 1 0
```

- Schärfung (Divisor 1):

```
 0, -1, 0
-1, 5, -1
 0, -1, 0
```

- Kantenerkennung (Divisor 1):

```
 0, -1, 0
-1, 4, -1
 0, -1, 0
```

- Prägung (Divisor 1):

```
-2, -1, 0
-1, 1, 1
 0, 1, 2
```

Beachten Sie, dass der Divisor für die meisten dieser Effekte den Wert 1 hat. Der Grund hierfür ist, dass die Summe der negativen Matrixwerte und der positiven Matrixwerte 1 ergibt (bzw. 0 für die Kantenerkennung, doch der Wert der `divisor`-Eigenschaft darf nicht 0 sein).

Verschiebungsmatrix-Filter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die DisplacementMapFilter-Klasse verwendet Pixelwerte eines BitmapData-Objekts (als Verschiebungsmatrixbild bezeichnet), um einen Verschiebungseffekt auf ein neues Objekt anzuwenden. Die Verschiebungsmatrix unterscheidet sich im Allgemeinen vom tatsächlichen Anzeigeobjekt bzw. der BitmapData-Instanz, auf die der Filter angewendet wird. Ein Verschiebungseffekt versetzt Pixel im gefilterten Bild – mit anderen Worten, sie werden um einen bestimmten Betrag von ihrer ursprünglichen Position weg verschoben. Mit diesem Filter kann ein verschobener, verzerrter oder mit einem Fleckenmuster versehener Effekt erzielt werden.

Die Position und der Betrag der Verschiebung eines bestimmten Pixels wird über den Farbwert der Verschiebungsmatrix berechnet. Wenn Sie diesen Filter verwenden, müssen Sie nicht nur das Matrixbild angeben, sondern auch die folgenden Werte zuweisen, um die Berechnung der Verschiebung aus dem Matrixbild steuern zu können:

- Matrixpunkt: Die Position auf dem gefilterten Bild, auf die die obere linke Ecke des Verschiebungsmatrix-Filters angewendet wird. Sie verwenden diese Option, wenn Sie den Filter nur auf einen Teil des Bilds anwenden möchten.
- X-Komponente: Legt fest, welcher Farbkanal des Matrixbilds sich auf die x-Position der Pixel auswirkt.
- Y-Komponente: Legt fest, welcher Farbkanal des Matrixbilds sich auf die y-Position der Pixel auswirkt.
- x-Skalierung: Ein Multiplikatorwert, mit dem festgelegt wird, wie stark die Verschiebung entlang der x-Achse ist.
- y-Skalierung: Ein Multiplikatorwert, mit dem festgelegt wird, wie stark die Verschiebung entlang der y-Achse ist.
- Filtermodus: Legt fest, was an den leeren Stellen ausgeführt werden soll, die aufgrund der verschobenen Pixel zurückbleiben. Folgende Optionen stehen zur Verfügung, die als Konstanten in der DisplacementMapFilterMode-Klasse definiert sind: Anzeigen der ursprünglichen Pixel (Filtermodus `IGNORE`), Verschieben der Pixel auf die andere Seite des Quellbilds (Filtermodus `WRAP`, die Standardeinstellung), Verwenden des nächsten verschobenen Pixels (Filtermodus `CLAMP`) oder Auffüllen leerer Flächen mit einer Farbe (Filtermodus `COLOR`).

Die Funktionsweise des Verschiebungsmatrix-Filters soll anhand eines einfachen Beispiels veranschaulicht werden. Im folgenden Code wird ein Bild geladen. Wenn es vollständig geladen ist, wird es auf der Bühne zentriert und ein Verschiebungsmatrix-Filter wird auf das Bild angewendet, wodurch die Pixel im gesamten Bild horizontal nach links verschoben werden.

```
import flash.display.BitmapData;
import flash.display.Loader;
import flash.events.MouseEvent;
import flash.filters.DisplacementMapFilter;
import flash.geom.Point;
import flash.net.URLRequest;

// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image3.jpg");
loader.load(url);
this.addChild(loader);

var mapImage:BitmapData;
var displacementMap:DisplacementMapFilter;

// This function is called when the image finishes loading.
function setupStage(event:Event):void
{
    // Center the loaded image on the Stage.
    loader.x = (stage.stageWidth - loader.width) / 2;
    loader.y = (stage.stageHeight - loader.height) / 2;

    // Create the displacement map image.
    mapImage = new BitmapData(loader.width, loader.height, false, 0xFF0000);

    // Create the displacement filter.
    displacementMap = new DisplacementMapFilter();
    displacementMap.mapBitmap = mapImage;
    displacementMap.mapPoint = new Point(0, 0);
    displacementMap.componentX = BitmapDataChannel.RED;
    displacementMap.scaleX = 250;
    loader.filters = [displacementMap];
}

loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setupStage);
```

Die zur Definition der Verschiebung verwendeten Eigenschaften lauten wie folgt:

- Matrix-Bitmap: Die Verschiebungs-Bitmap ist eine neue BitmapData-Instanz, die vom Code erstellt wurde. Ihre Abmessungen entsprechen denen des geladenen Bilds (somit wird die Verschiebung auf das gesamte Bild angewendet). Es wird mit festen roten Pixeln aufgefüllt.
- Matrixpunkt: Dieser Wert wird auf den Punkt 0,0 eingestellt – auch dies führt dazu, dass die Verschiebung auf das gesamte Bild angewendet wird.
- X-Komponente: Dieser Wert wird auf die Konstante `BitmapDataChannel.RED` gesetzt. Dies bedeutet, dass der Rotwert der Matrix-Bitmap festlegt, um welchen Abstand die Pixel auf der x-Achse verschoben (versetzt) werden.
- x-Skalierung: Dieser Wert wird auf 250 festgelegt. Beim vollständigen Ausmaß der Verschiebung (von einem vollständig rotem Matrixbild) wird das Bild nur geringfügig verschoben (etwa um die Hälfte eines Pixels). Wenn dieser Wert also auf 1 gesetzt ist, wird das Bild daher um nur 0,5 Pixel horizontal verschoben. Bei einer Einstellung von 250 wird das Bild um etwa 125 Pixel verschoben.

Bei diesen Einstellungen werden die Pixel des gefilterten Bilds um 250 Pixel nach links verschoben. Die Richtung (links oder rechts) und der Betrag der Verschiebung basieren auf dem Farbwert der Pixel im Matrixbild. Im Prinzip durchläuft der Filter nacheinander die Pixel des gefilterten Bilds (zumindest die Pixel in dem Bereich, auf den der Filter angewendet wurde, d. h. in diesem Fall alle Pixel), und führt Folgendes mit jedem Pixel aus:

1. Er sucht die entsprechenden Pixel im Matrixbild. Wenn der Filter z. B. die Verschiebung für das Pixel in der linken oberen Ecke des gefilterten Bilds berechnet, sucht er nach dem entsprechenden Pixel in der linken oberen Ecke des Matrixbilds.
2. Er ermittelt den Wert des angegebenen Farbkanals im Matrixpixel. In diesem Fall ist der Farbkanal der X-Komponente der Rotkanal, also sucht der Filter nach dem Wert des Rotkanals im Matrixbild an dem betreffenden Pixel. Da das Matrixbild vollständig rot ist, lautet der Wert für den Rotkanal des Pixels 0xFF bzw. 255. Dieser Wert wird für die Verschiebung verwendet.
3. Er vergleicht den Verschiebungswert mit dem Mittelwert (127, die Hälfte zwischen 0 und 255). Ist der Verschiebungswert kleiner als der Mittelwert, werden die Pixel in positiver Richtung verschoben (bei einer x-Verschiebung nach rechts; bei einer y-Verschiebung nach unten). Ist der Verschiebungswert größer als der Mittelwert (wie in diesem Beispiel), werden die Pixel in negativer Richtung verschoben (bei einer x-Verschiebung nach links; bei einer y-Verschiebung nach oben). Um genau zu sein, subtrahiert der Filter den Verschiebungswert von 127, und das Ergebnis (positiv oder negativ) ist der relative Betrag der Verschiebung, der letztlich angewendet wird.
4. Schließlich berechnet der Filter den tatsächlichen Betrag der Verschiebung, indem ermittelt wird, welchen Prozentsatz der vollständigen Verschiebung die relative Verschiebung darstellt. In diesem Fall bedeutet vollständig rot eine Verschiebung um 100 %. Dieser Prozentsatz wird dann mit dem Wert der x-Skalierung oder der y-Skalierung multipliziert, um die Anzahl der Pixel der angewendeten Verschiebung zu berechnen. In diesem Beispiel wird der Betrag der Verschiebung über 100 % mal einem Multiplikator von 250 berechnet – in etwa 125 Pixel nach links.

Da keine Werte für y-Komponente und y-Skalierung angegeben wurden, werden die Standardwerte verwendet (die keine Verschiebung verursachen). Aus diesem Grund wird das Bild nicht in vertikaler Richtung verschoben.

In diesem Beispiel wird die Standardeinstellung `WRAP` für den Filtermodus verwendet, sodass der leere Bereich auf der rechten Seite beim Verschieben der Pixel nach links mit den Pixeln gefüllt wird, die über den linken Rand des Bilds hinausgeschoben wurden. Sie können ein wenig mit diesem Wert experimentieren, um die unterschiedlichen Effekte zu betrachten. Wenn Sie beispielsweise die folgende Zeile zu dem Codeteil hinzufügen, in dem die Eigenschaften der Verschiebung angegeben werden (vor der Zeile `loader.filters = [displacementMap]`), sieht das Bild so aus, als wäre es über die Bühne geschmiert:

```
displacementMap.mode = DisplacementMapFilterMode.CLAMP;
```

Im folgenden komplexeren Codebeispiel wird ein Verschiebungsmatrixfilter verwendet, um den Effekt eines Vergrößerungsglases über dem Bild zu erzeugen:

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.display.BitmapDataChannel;
import flash.display.GradientType;
import flash.display.Loader;
import flash.display.Shape;
import flash.events.MouseEvent;
import flash.filters.DisplacementMapFilter;
import flash.filters.DisplacementMapFilterMode;
import flash.geom.Matrix;
import flash.geom.Point;
import flash.net.URLRequest;

// Create the gradient circles that will together form the
// displacement map image
var radius:uint = 50;

var type:String = GradientType.LINEAR;
var redColors:Array = [0xFF0000, 0x000000];
var blueColors:Array = [0x0000FF, 0x000000];
var alphas:Array = [1, 1];
var ratios:Array = [0, 255];
var xMatrix:Matrix = new Matrix();
xMatrix.createGradientBox(radius * 2, radius * 2);
var yMatrix:Matrix = new Matrix();
yMatrix.createGradientBox(radius * 2, radius * 2, Math.PI / 2);

var xCircle:Shape = new Shape();
xCircle.graphics.lineStyle(0, 0, 0);
xCircle.graphics.beginGradientFill(type, redColors, alphas, ratios, xMatrix);
xCircle.graphics.drawCircle(radius, radius, radius);

var yCircle:Shape = new Shape();
yCircle.graphics.lineStyle(0, 0, 0);
yCircle.graphics.beginGradientFill(type, blueColors, alphas, ratios, yMatrix);
yCircle.graphics.drawCircle(radius, radius, radius);

// Position the circles at the bottom of the screen, for reference.
this.addChild(xCircle);
xCircle.y = stage.stageHeight - xCircle.height;
this.addChild(yCircle);
yCircle.y = stage.stageHeight - yCircle.height;
yCircle.x = 200;

// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

// Create the map image by combining the two gradient circles.
var map:BitmapData = new BitmapData(xCircle.width, xCircle.height, false, 0x7F7F7F);
map.draw(xCircle);
var yMap:BitmapData = new BitmapData(yCircle.width, yCircle.height, false, 0x7F7F7F);
yMap.draw(yCircle);
map.copyChannel(yMap, yMap.rect, new Point(0, 0), BitmapDataChannel.BLUE,
BitmapDataChannel.BLUE);
```

```
yMap.dispose();

// Display the map image on the Stage, for reference.
var mapBitmap:Bitmap = new Bitmap(map);
this.addChild(mapBitmap);
mapBitmap.x = 400;
mapBitmap.y = stage.stageHeight - mapBitmap.height;

// This function creates the displacement map filter at the mouse location.
function magnify():void
{
    // Position the filter.
    var filterX:Number = (loader.mouseX) - (map.width / 2);
    var filterY:Number = (loader.mouseY) - (map.height / 2);
    var pt:Point = new Point(filterX, filterY);
    var xyFilter:DisplacementMapFilter = new DisplacementMapFilter();
    xyFilter.mapBitmap = map;
    xyFilter.mapPoint = pt;
    // The red in the map image will control x displacement.
    xyFilter.componentX = BitmapDataChannel.RED;
    // The blue in the map image will control y displacement.
    xyFilter.componentY = BitmapDataChannel.BLUE;
    xyFilter.scaleX = 35;
    xyFilter.scaleY = 35;
    xyFilter.mode = DisplacementMapFilterMode.IGNORE;
    loader.filters = [xyFilter];
}

// This function is called when the mouse moves. If the mouse is
// over the loaded image, it applies the filter.
function moveMagnifier(event:MouseEvent):void
{
    if (loader.hitTestPoint(loader.mouseX, loader.mouseY))
    {
        magnify();
    }
}
loader.addEventListener(MouseEvent.MOUSE_MOVE, moveMagnifier);
```

Im Code werden zunächst zwei Farbverlaufkreise erstellt, die kombiniert werden, um die Verschiebungsmatrix zu bilden. Der rote Kreis erzeugt die Verschiebung auf der x-Achse (`xyFilter.componentX = BitmapDataChannel.RED`) und der blaue Kreis die Verschiebung auf der y-Achse (`xyFilter.componentY = BitmapDataChannel.BLUE`). Damit Sie eine bessere Vorstellung davon haben, wie die Verschiebungsmatrix aussieht, werden im Code zusätzlich die Originalkreise sowie der kombinierte Kreis, der als Matrixbild dient, am unteren Bildschirmrand dargestellt.

Dann lädt der Code ein Bild. Wenn sich die Maus bewegt, wird der Verschiebungsmatrixfilter auf den Bereich des Bilds angewendet, der sich unter der Maus befindet. Die Farbverlaufkreise, die als Verschiebungsmatrixfilter verwendet werden, führen dazu, dass der verschobene Bereich vom Zeiger weg verteilt wird. Beachten Sie, dass die grauen Bereiche der Verschiebungsmatrix keine Verschiebung verursachen. Die Farbe Grau hat den Wert `0x7F7F7F`. Die Blau- und Rotkanäle dieses Grautons entsprechen exakt dem mittleren Ton dieser Farbkanäle, somit findet im grauen Bereich des Matrixbilds keine Verschiebung statt. Entsprechend gibt es im Mittelpunkt des Kreises keine Verschiebung. Obwohl die Farbe dort nicht grau ist, sind der Blaukanal und Rotkanal dieser Farbe identisch mit dem Blau- und Rotkanal eines mittleren Graus, und da nur Blau und Rot zu einer Verschiebung führen, findet hier keine Verschiebung statt.

Shader-Filter

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Mit der ShaderFilter-Klasse können Sie einen benutzerdefinierten Filtereffekt anwenden, der als Pixel Bender-Shader definiert ist. Da der Filtereffekt als Pixel Bender-Shader geschrieben wird, kann der Effekt vollständig angepasst werden. Der gefilterte Inhalt wird als Bildeingabe an den Shader übergeben, und das Ergebnis des Shader-Vorgangs wird zum Filterergebnis.

Hinweis: *Der Shader-Filter ist in ActionScript ab Flash Player 10 und Adobe AIR 1.5 verfügbar.*

Um einen Shader-Filter auf ein Objekt anzuwenden, erstellen Sie zunächst eine Shader-Instanz, die den verwendeten Pixel Bender-Shader darstellt. Informationen zum Erstellen einer Shader-Instanz und zum Angeben von Eingabebild- und Parameterwerten finden Sie unter „Arbeiten mit Pixel Bender-Shadern“ auf Seite 321.

Beim Verwenden eines Shaders als Filter sind drei wichtige Dinge zu beachten:

- Der Shader muss mindestens ein Eingabebild akzeptieren.

- Das gefilterte Objekt (das Anzeigeobjekt oder BitmapData-Objekt, auf das der Filter angewendet wird) wird als erster Eingabebildwert an den Shader übergeben. Geben Sie daher den Wert für die erste Bildeingabe nicht manuell an.
- Wenn der Shader mehr als ein Eingabebild definiert, müssen die zusätzlichen Eingaben manuell vorgenommen werden (durch Festlegen der `input`-Eigenschaft von ShaderInput-Instanzen, die zur Shader-Instanz gehören).

Sobald ein Shader-Objekt für den Shader vorliegt, können Sie eine ShaderFilter-Instanz erstellen. Dies ist das eigentliche Filterobjekt, das Sie wie jeden anderen Filter verwenden. Um einen ShaderFilter zu erstellen, der ein Shader-Objekt verwendet, rufen Sie den `ShaderFilter()`-Konstruktor auf und übergeben Sie das Shader-Objekt als Argument wie in diesem Beispiel gezeigt:

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
```

Ein ausführliches Beispiel für die Verwendung eines Shader-Filters finden Sie unter „Verwenden eines Shaders als Filter“ auf Seite 339.

Beispiel für das Filtern von Anzeigeobjekten: Filter Workbench

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Anwendung „Filter Workbench“ bietet eine Benutzeroberfläche, über die verschiedene Filter auf Bilder und andere visuelle Inhalte angewendet werden können. Außerdem wird der resultierende Code angezeigt, mit dem die gleichen Effekte in ActionScript erzeugt werden können. Mit diesem Tool können Sie die Effekte verschiedener Filter testen. Zusätzlich werden mit dieser Anwendung die folgenden Techniken verdeutlicht:

- Erstellen von Instanzen verschiedener Filter
- Anwenden mehrerer Filter auf ein Anzeigeobjekt

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „Filter Workbench“ befinden sich im Ordner „Samples/FilterWorkbench“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
com/example/programmingas3/filterWorkbench/FilterWorkbenchController.as	Klasse mit den Hauptfunktionen der Anwendung, einschließlich Ändern des Inhalts, auf den Filter angewendet werden, und Anwenden von Filtern auf Inhalte.
com/example/programmingas3/filterWorkbench/IFilterFactory.as	Schnittstelle zur Definition allgemeiner Methoden, die von allen Factory-Klassen für Filter implementiert werden. Über diese Schnittstelle werden die gemeinsamen Funktionen definiert, die von der FilterWorkbenchController-Klasse zum Interagieren mit den einzelnen Factory-Klassen für Filter verwendet werden.
Im Ordner com/example/programmingas3/filterWorkbench/: BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	Gruppe von Klassen, die die IFilterFactory-Schnittstelle implementieren. Jede dieser Klassen enthält die Funktionen zum Erstellen eines bestimmten Filtertyps und zum Festlegen von entsprechenden Werten. In der Anwendung werden in den Eigenschaftenbedienfeldern mithilfe dieser Factory-Klassen Instanzen der jeweiligen Filter erstellt, die dann von der FilterWorkbenchController-Klasse abgerufen und auf den Bildinhalt angewendet werden.
com/example/programmingas3/filterWorkbench/IFilterPanel.as	Schnittstelle zur Definition allgemeiner Methoden, die von Klassen implementiert werden, die die zum Bearbeiten von Filterwerten verwendeten Benutzerobenflächen-Bedienfelder definieren.
com/example/programmingas3/filterWorkbench/ColorStringFormatter.as	Dienstprogrammklasse, die eine Methode zum Konvertieren von numerischen Farbwerten in das hexadezimale Stringformat enthält.
com/example/programmingas3/filterWorkbench/GradientColor.as	Klasse, die als Wertobjekt dient und in der die jeder Farbe in GradientBevelFilter und GradientGlowFilter zugeordneten drei Werte (Farbe, Alpha und Verhältnis) in einem Objekt kombiniert werden.
Benutzeroberfläche (Flex)	
FilterWorkbench.mxml	Die Hauptdatei für die Benutzeroberfläche der Anwendung.
flexapp/FilterWorkbench.as	Klasse mit den Funktionen für die Benutzeroberfläche der Hauptanwendung. Diese Klasse wird als CodeBehind-Klasse für die MXML-Datei der Anwendung verwendet.

Datei	Beschreibung
Im Ordner flexapp/filterPanels: BevelPanel.mxml BlurPanel.mxml ColorMatrixPanel.mxml ConvolutionPanel.mxml DropShadowPanel.mxml GlowPanel.mxml GradientBevelPanel.mxml GradientGlowPanel.mxml	Gruppe von MXML-Komponenten mit den Funktionen für die einzelnen Bedienfelder, in denen die Optionen für den jeweiligen Filter festgelegt werden.
flexapp/ImageContainer.as	Ein Anzeigeobjekt, das auf dem Bildschirm als Container für das geladene Bild dient.
flexapp/controls/BGColorCellRenderer.as	Benutzerdefinierte CellRenderer-Klasse zum Ändern der Hintergrundfarbe einer Zelle in der DataGrid-Komponente.
flexapp/controls/QualityComboBox.as	Benutzerdefiniertes Steuerelement, das ein Kombinationsfeld definiert, das für die Qualitätseinstellungen in verschiedenen Filter-Bedienfeldern verwendet werden kann.
flexapp/controls/TypeComboBox.as	Benutzerdefiniertes Steuerelement, das ein Kombinationsfeld definiert, das für die Typeinstellungen in verschiedenen Filter-Bedienfeldern verwendet werden kann.
Benutzeroberfläche (Flash)	
FilterWorkbench.fla	Die Hauptdatei für die Benutzeroberfläche der Anwendung.
flashapp/FilterWorkbench.as	Klasse mit den Funktionen für die Benutzeroberfläche der Hauptanwendung. Diese Klasse wird als Dokumentklasse für die FLA-Datei der Anwendung verwendet.
Im Ordner flashapp/filterPanels: BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	Gruppe von Klassen mit den Funktionen für die einzelnen Bedienfelder, in denen die Optionen für den jeweiligen Filter festgelegt werden. Jeder Klasse ist in der Bibliothek der FLA-Datei für die Hauptanwendung auch ein MovieClip-Symbol zugeordnet, dessen Name dem Namen der Klasse entspricht (das Symbol „BlurPanel“ ist beispielsweise mit der in „BlurPanel.as“ definierten Klasse verknüpft). Die Komponenten der Benutzeroberfläche werden in diesen Symbolen positioniert und benannt.
flashapp/ImageContainer.as	Ein Anzeigeobjekt, das auf dem Bildschirm als Container für das geladene Bild dient.
flashapp/BGColorCellRenderer.as	Benutzerdefinierte CellRenderer-Klasse zum Ändern der Hintergrundfarbe einer Zelle in der DataGrid-Komponente.

Datei	Beschreibung
flashapp/ButtonCellRenderer.as	Benutzerdefinierte CellRenderer-Klasse zum Einfügen einer Button-Komponente in eine Zelle der DataGrid-Komponente.
Gefilterter Bildinhalt	
com/example/programmingas3/filterWorkbench/ImageType.as	Diese Klasse dient als Wertobjekt und enthält den Typ und die URL einer einzelnen Bilddatei. Für dieses Objekt können in der Anwendung Filter geladen und angewendet werden. Die Klasse enthält auch mehrere Konstanten, die die tatsächlich verfügbaren Bilddateien darstellen.
images/sampleAnimation.swf, images/sampleImage1.jpg, images/sampleImage2.jpg	Bilder und andere visuelle Inhalte, auf die in der Anwendung Filter angewendet werden.

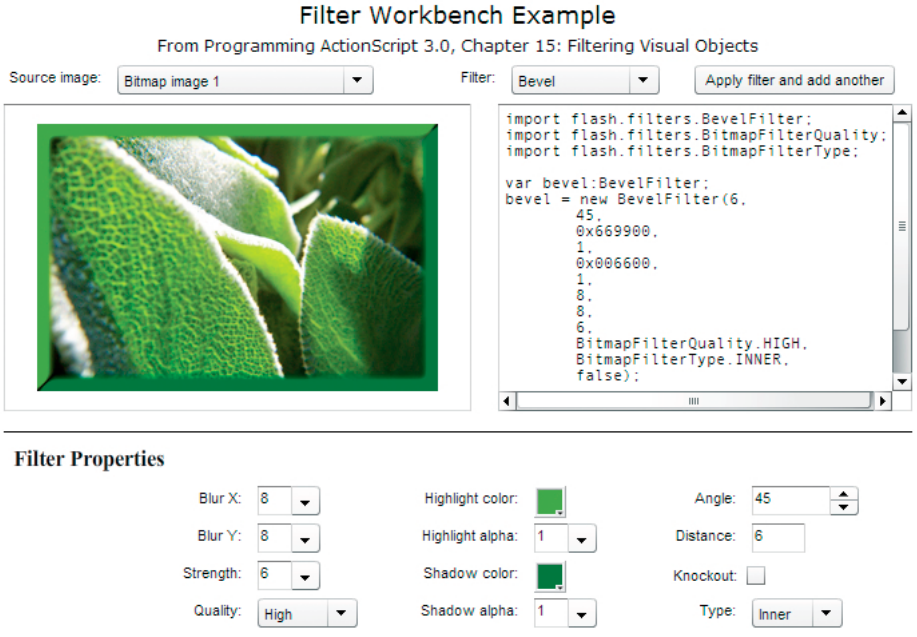
Experimentieren mit ActionScript-Filtern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der Anwendung „Filter Workbench“ können Sie verschiedene Filtereffekte testen und den entsprechenden ActionScript-Code für die jeweiligen Effekte erzeugen. In der Anwendung haben Sie die Auswahl zwischen drei verschiedenen Dateien mit grafischem Inhalt, einschließlich Bitmapgrafiken und einer mit Flash erstellten Animation. Zudem können Sie acht verschiedene ActionScript-Filter entweder einzeln oder in Kombination mit anderen Filtern auf das ausgewählte Bild anwenden. Die Anwendung enthält die folgenden Filter:

- Geschliffen (flash.filters.BevelFilter)
- Weichzeichnen (flash.filters.BlurFilter)
- Farbmatrix (flash.filters.ColorMatrixFilter)
- Convolution (flash.filters.ConvolutionFilter)
- Schlagschatten (flash.filters.DropShadowFilter)
- Glühen (flash.filters.GlowFilter)
- Farbverlauf-Geschliffen (flash.filters.GradientBevelFilter)
- Farbverlauf-Glühen (flash.filters.GradientGlowFilter)

Wenn ein Benutzer ein Bild und einen auf das Bild anzuwendenden Filter ausgewählt hat, wird ein Bedienfeld mit Steuerelementen angezeigt, über die die spezifischen Eigenschaften des ausgewählten Filters festgelegt werden können. In der folgenden Abbildung ist beispielsweise der Geschliffen-Filter ausgewählt:



Wenn der Benutzer die Filtereigenschaften ändert, wird die Vorschau in Echtzeit aktualisiert. Der Benutzer kann auch mehrere Filter anwenden, indem er einen Filter anpasst, auf die Schaltfläche „Anwenden“ klickt, dann einen anderen Filter anpasst, auf die Schaltfläche „Anwenden“ klickt usw.

Es folgt die Beschreibung einiger Funktionen und Beschränkungen bei einzelnen Bedienfeldern für Filter in der Anwendung:

- Der Farbmatrixfilter enthält mehrere Steuerelemente zur direkten Bearbeitung allgemeiner Bildeigenschaften, z. B. Helligkeit, Kontraste, Sättigung und Farbton. Zusätzlich können benutzerdefinierte Farbmatrixwerte angegeben werden.
- Der Convolution-Filter, der nur bei Verwendung von ActionScript verfügbar ist, enthält mehrere häufig verwendete Convolution-Matrixwerte. Zudem können benutzerdefinierte Werte angegeben werden. Zwar kann bei der ConvolutionFilter-Klasse eine Matrix beliebiger Größe verwendet werden, in der Anwendung „Filter Workbench“ wird jedoch eine feste 3x3-Matrix verwendet (die am häufigsten verwendete Filtergröße).
- Der Verschiebungsmatrixfilter und der Shader-Filter, die nur in ActionScript verfügbar sind, stehen in der Anwendung „Filter Workbench“ nicht zur Verfügung.

Erstellen von Filterinstanzen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Anwendung „Filter Workbench“ enthält für jeden der verfügbaren Filter je eine Klasse, die im jeweiligen Bedienfeld zum Erstellen der Filterinstanz verwendet wird. Wenn ein Benutzer einen Filter auswählt, wird mit dem ActionScript-Code, der dem Bedienfeld für den Filter zugeordnet ist, eine Instanz der entsprechenden Factory-Klasse für Filter erstellt. (Diese Klassen werden als *Factory-Klassen* bezeichnet, da sie zum Erstellen von Instanzen anderer Objekte dienen, ähnlich einer Fabrik (engl. „factory“), in der einzelne Produkte hergestellt werden.)

Bei jeder Änderung eines Eigenschaftswerts im Bedienfeld wird durch den Code des Bedienfelds die entsprechende Methode der Factory-Klasse aufgerufen. Jede Factory-Klasse enthält spezifische Methoden, mit denen im Bedienfeld die entsprechende Filterinstanz erstellt wird. Wenn der Benutzer beispielsweise den Weichzeichnen-Filter auswählt, wird in der Anwendung eine BlurFactory-Instanz erstellt. Die BlurFactory-Klasse enthält eine `modifyFilter()`-Methode, die drei Parameter akzeptiert: `blurX`, `blurY` und `quality`, die zusammen für die Erstellung der gewünschten BlurFilter-Instanz verwendet werden:

```
private var _filter:BlurFilter;

public function modifyFilter(blurX:Number = 4, blurY:Number = 4, quality:int = 1):void
{
    _filter = new BlurFilter(blurX, blurY, quality);
    dispatchEvent(new Event(Event.CHANGE));
}
```

Wenn der Benutzer dagegen den Convolution-Filter auswählt, der eine viel größere Flexibilität zulässt, ist eine größere Anzahl festzulegender Eigenschaften verfügbar. In der ConvolutionFactory-Klasse wird der folgende Code aufgerufen, wenn der Benutzer im Bedienfeld einen anderen Wert auswählt:

```
private var _filter:ConvolutionFilter;

public function modifyFilter(matrixX:Number = 0,
                                            matrixY:Number = 0,
                                            matrix:Array = null,
                                            divisor:Number = 1.0,
                                            bias:Number = 0.0,
                                            preserveAlpha:Boolean = true,
                                            clamp:Boolean = true,
                                            color:uint = 0,
                                            alpha:Number = 0.0):void
{
    _filter = new ConvolutionFilter(matrixX, matrixY, matrix, divisor, bias, preserveAlpha,
clamp, color, alpha);
    dispatchEvent(new Event(Event.CHANGE));
}
```

Bei jeder Änderung der Werte für den Filter löst das Factory-Objekt ein `Event.CHANGE`-Ereignis aus, um die Listener zu benachrichtigen, dass die Werte des Filters geändert wurden. Die FilterWorkbenchController-Klasse, mit der Filter auf den zu filternden Inhalt angewendet werden, überwacht das Eintreten dieses Ereignisses und ermittelt den Zeitpunkt, zu dem eine neue Kopie des Filters abgerufen und der Filter erneut auf den zu filternden Inhalt angewendet werden muss.

An die FilterWorkbenchController-Klasse müssen nicht die spezifischen Einzelheiten der einzelnen Factory-Klassen für Filter übergeben werden. Es muss lediglich angegeben werden, dass ein Filter geändert wurde, und die Klasse muss auf eine Kopie des Filters zugreifen können. Dazu enthält die Anwendung die IFilterFactory-Schnittstelle, über die das erforderliche Verhalten der Factory-Klassen für Filter definiert wird, sodass die entsprechenden Vorgänge durch die FilterWorkbenchController-Instanz durchgeführt werden können. Die IFilterFactory-Schnittstelle definiert die `getFilter()`-Methode, die in der FilterWorkbenchController-Klasse verwendet wird:

```
function getFilter():BitmapFilter;
```

Beachten Sie, dass in der Definition der `getFilter()`-Methode angegeben ist, dass kein bestimmter Filtertyp, sondern eine BitmapFilter-Instanz zurückgegeben wird. Mit der BitmapFilter-Klasse wird kein bestimmter Filtertyp definiert. Bei der BitmapFilter-Klasse handelt es sich vielmehr um die Basisklasse, die allen Filterklassen zugrunde liegt. Jede Factory-Klasse für Filter definiert eine spezifische Implementierung der `getFilter()`-Methode, in der ein Verweis auf das durch die Klasse erstellte Filterobjekt zurückgegeben wird. Es folgt der Quellcode der ConvolutionFactory-Klasse in gekürzter Form:

```
public class ConvolutionFactory extends EventDispatcher implements IFilterFactory
{
    // ------- Private vars -------
    private var _filter:ConvolutionFilter;
    ...
    // ------- IFilterFactory implementation -------
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}
```

In der Implementierung der `getFilter()`-Methode durch die ConvolutionFactory-Klasse wird eine ConvolutionFilter-Instanz zurückgegeben, auch wenn dieser Instanztyp für Objekte, die `getFilter()` aufrufen, u. U. unbekannt ist. Entsprechend der Definition der `getFilter()`-Methode durch die ConvolutionFactory-Klasse kann eine beliebige BitmapFilter-Instanz zurückgegeben werden, die eine Instanz einer der ActionScript-Filterklassen ist.

Anwenden von Filtern auf Anzeigeobjekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie bereits erläutert wurde, wird in der Anwendung „Filter Workbench“ eine Instanz der FilterWorkbenchController-Klasse (im Folgenden als „Controller-Instanz“ bezeichnet) verwendet, um Filter auf das ausgewählte Anzeigeobjekt anzuwenden. Damit über die Controller-Instanz Filter angewendet werden können, muss zunächst festgelegt werden, auf welches Bild oder welchen grafischen Inhalt der entsprechende Filter angewendet werden soll. Wenn der Benutzer ein Bild auswählt, wird in der Anwendung die `setFilterTarget()`-Methode der FilterWorkbenchController-Klasse aufgerufen und eine der in der ImageType-Klasse definierten Konstanten übergeben:

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, targetLoadComplete);
    ...
}
```

Anhand dieser Informationen wird die angegebene Datei durch die Controller-Instanz geladen und nach dem Laden in der Instanzvariable `_currentTarget` gespeichert:

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

Wenn der Benutzer einen Filter auswählt, wird die `setFilter()`-Methode der Controller-Instanz mit einem Verweis auf das relevante Factory-Objekt für Filter aufgerufen, das dann in der Instanzvariable `_filterFactory` gespeichert wird.

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

Wie zuvor beschrieben, erkennt die Controller-Instanz den spezifischen Datentyp der angegebenen Factory-Instanz für Filter nicht. Sie erkennt lediglich, dass das Objekt die IFilterFactory-Instanz implementiert, d. h., dass sie über eine `getFilter()`-Methode verfügt und ein `change`-Ereignis (`Event.CHANGE`) auslöst, wenn der Filter geändert wird.

Wenn der Benutzer die Eigenschaften eines Filters im Bedienfeld für den Filter ändert, erkennt die Controller-Instanz die Änderung des Filters durch das `change`-Ereignis der Factory-Klasse für Filter, bei dem die `filterChange()`-Methode der Controller-Instanz aufgerufen wird. Diese Methode ruft wiederum die `applyTemporaryFilter()`-Methode auf:

```
private function filterChange(event:Event):void
{
    applyTemporaryFilter();
}

private function applyTemporaryFilter():void
{
    var currentFilter:BitmapFilter = _filterFactory.getFilter();

    // Add the current filter to the set temporarily
    _currentFilters.push(currentFilter);

    // Refresh the filter set of the filter target
    _currentTarget.filters = _currentFilters;

    // Remove the current filter from the set
    // (This doesn't remove it from the filter target, since
    // the target uses a copy of the filters array internally.)
    _currentFilters.pop();
}
```

Das Anwenden des Filters auf das Anzeigeobjekt erfolgt in der `applyTemporaryFilter()`-Methode. Die Controller-Instanz ruft zunächst einen Verweis auf das Filterobjekt ab, indem sie die `getFilter()`-Methode der Factory-Klasse für Filter aufruft.

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

Die Controller-Instanz verfügt über die Array-Instanzvariable `_currentFilters`, in der alle Filter gespeichert sind, die auf das Anzeigeobjekt angewendet wurden. Der neu aktualisierte Filter wird dann diesem Array hinzugefügt:

```
_currentFilters.push(currentFilter);
```

Anschließend wird das Filter-Array der `filters`-Eigenschaft des Anzeigeobjekts zugewiesen, mit der die Filter dann auf das Bild angewendet werden:

```
_currentTarget.filters = _currentFilters;
```

Da der zuletzt hinzugefügte Filter bis zum endgültigen Anwenden nur getestet wird, darf er nicht dauerhaft auf das Anzeigeobjekt angewendet werden. Deshalb wird er wieder aus dem `_currentFilters`-Array entfernt:

```
_currentFilters.pop();
```

Das Entfernen des Filters aus dem Array wirkt sich nicht auf das gefilterte Anzeigeobjekt aus, da im Anzeigeobjekt eine Kopie des Filter-Arrays erstellt wird, wenn es der `filters`-Eigenschaft zugewiesen wird. Dieses interne Array wird dann anstelle des Original-Arrays verwendet. Aus diesem Grund wirken sich Änderungen, die am Filter-Array vorgenommen werden, nur dann auf das Anzeigeobjekt aus, wenn das Array erneut der `filters`-Eigenschaft des Anzeigeobjekts zugewiesen wird.

Kapitel 15: Arbeiten mit Pixel Bender-Shadern

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Mit dem Adobe Pixel Bender Toolkit können Entwickler Shader schreiben, die grafische Effekte erzeugen, und andere Bild- sowie Datenverarbeitungsaufgaben durchführen. Der Pixel Bender-Bytecode kann in ActionScript ausgeführt werden, um den Effekt auf Bilddaten oder visuelle Inhalte anzuwenden. Durch die Verwendung von Pixel Bender-Shadern in ActionScript haben Sie die Möglichkeit, benutzerdefinierte visuelle Effekte zu erstellen und Datenverarbeitungsaufgaben auszuführen, die über die integrierten Funktionen von ActionScript hinausgehen.

Hinweis: *Pixel Bender-Unterstützung ist ab Flash Player 10 und Adobe AIR 1.5 verfügbar. Pixel Bender-Mischungen, -Filter und -Füllungen werden beim GPU-Rendern nicht unterstützt.*

Verwandte Hilfethemen

Adobe Pixel Bender Technology Center

Pixel Bender Developer's Guide

Pixel Bender Reference

flash.display.Shader

flash.filters.ShaderFilter

Pixel Bender basics for Flash

Pixel Bender basics for Flex

Grundlagen von Pixel Bender-Shadern

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Adobe Pixel Bender ist eine Programmiersprache, mit der Bildinhalte erstellt oder manipuliert werden können. Mit Pixel Bender können Sie einen Kernel erstellen, der auch als Shader bezeichnet wird. Der Shader definiert eine einzelne Funktion, die an jedem einzelnen der Pixel eines Bildes ausgeführt wird. Das Ergebnis eines jeden Aufrufs der Funktion ist die Ausgabefarbe an der jeweiligen Pixelkoordinate im Bild. Eingabebilder und Parameterwerte können zur Anpassung des Vorgangs angegeben werden. Bei jeder Ausführung des Shaders bleiben Eingabe- und Parameterwerte konstant. Es ändert sich lediglich die Koordinate des Pixels, dessen Farbe das Ergebnis des Funktionsaufrufs ist.

Wenn möglich, wird die Shader-Funktion für mehrere Ausgabepixelkoordinaten parallel aufgerufen. Dies verbessert die Leistung des Shaders und beschleunigt die Verarbeitung.

In ActionScript lassen sich drei Effekttypen mithilfe eines Shaders schnell erstellen:

- Zeichnungsfüllung
- Mischmodus

- Filter

Ein Shader kann auch im Standalone-Modus ausgeführt werden. Im Standalone-Modus wird die beabsichtigte Verwendung nicht im Voraus angegeben, sondern es wird direkt auf das Ergebnis des Shaders zugegriffen. Auf die Ergebnisse kann als Bilddaten oder Binär- oder Zahlendaten zugegriffen werden. Bei den Daten muss es sich nicht um Bilddaten handeln. So können Sie einem Shader auch mit einer Gruppe von Daten als Eingabe versehen. Der Shader verarbeitet die Daten und Sie können auf die vom Shader zurückgegebenen Ergebnisdaten zugreifen.

Pixel Bender-Unterstützung ist ab Flash Player 10 und Adobe AIR 1.5 verfügbar. Pixel Bender-Mischungen, -Filter und -Füllungen werden beim GPU-Rendern nicht unterstützt. Auf Mobilgeräten können Pixel Bender-Shader mit dem CPU-Rendern ausgeführt werden. Die Leistung ist jedoch nicht so gut wie bei Desktopcomputern. Viele Shaderprogramme werden möglicherweise nur mit einer Geschwindigkeit von wenigen Bildern pro Sekunde ausgeführt.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe aufgeführt, die Ihnen beim Erstellen und Verwenden von Pixel Bender-Shadern begegnen:

Kernel Für Pixel Bender ist ein Kernel dasselbe wie ein Shader. Bei der Verwendung von Pixel Bender definiert Ihr Code einen Kernel, der eine einzelne Funktion definiert, die an jedem der Pixel eines Bildes separat ausgeführt wird.

Pixel Bender-Bytecode Wenn ein Pixel Bender-Kernel kompiliert wird, wird er in einen Pixel Bender-Bytecode umgewandelt. Der Bytecode wird zur Laufzeit ausgeführt.

Pixel Bender-Sprache Die Programmiersprache, die zur Erstellung eines Pixel Bender-Kernels verwendet wird.

Pixel Bender-Toolkit Die Anwendung, mit der eine Pixel Bender-Bytecodedatei aus Pixel Bender-Quellcode erstellt wird. Mit dem Toolkit können Sie Pixel Bender-Quellcode schreiben, testen und kompilieren.

Shader Für die Zwecke dieses Dokuments ist ein Shader ein Satz von Funktionen, die in der Pixel Bender-Sprache geschrieben wurden. Der Code eines Shaders erzeugt visuelle Effekte oder führt eine Berechnung durch. In beiden Fällen gibt der Shader eine Gruppe von Daten zurück (in der Regel die Pixel eines Bildes). Der Shader führt denselben Vorgang für jeden Datenpunkt aus. Der einzige Unterschied liegt in den Koordinaten des Ausgabepixels. Der Shader wird nicht in ActionScript geschrieben. Er wird in der Pixel Bender-Sprache geschrieben und in Pixel Bender-Bytecode kompiliert. Er kann während der Kompilierung in eine SWF-Datei eingebettet oder zur Laufzeit als externe Datei geladen werden. In beiden Fällen wird in ActionScript durch die Erstellung eines Shader-Objekts, das mit dem Shader-Bytecode verknüpft wird, auf ihn zugegriffen.

Shader-Eingabe Eine komplexe Eingabe, in der Regel Bitmap-Bilddaten, die dem Shader für seine Berechnungen zur Verfügung gestellt werden. Für jede in einem Shader definierte Eingabevariable wird ein einzelner Wert (also ein einzelnes Bild oder ein Satz von Binärdaten) für die gesamte Ausführung des Shaders verwendet.

Shader-Parameter Ein einzelner Wert oder ein eingeschränkter Satz von Werten, die dem Shader für seine Berechnungen zur Verfügung gestellt werden. Jeder Parameterwert wird für eine einzelne Shader-Ausführung definiert. Dieser Wert wird dann während der gesamten Ausführung des Shaders verwendet.

Verwenden der Codebeispiele

Vielleicht möchten Sie die bereitgestellten Codebeispiele testen. Dies beinhaltet das Ausführen des Codes und das Betrachten der Ergebnisse in der erstellten SWF-Datei. In allen Beispielen werden Inhalte mit der Zeichnungs-API erstellt, welche den Shader-Effekt verwendet oder durch in verändert wird.

Der Großteil des Beispielcodes besteht aus zwei Teilen. Zum einen aus dem Pixel Bender-Quellcode für den Shader, der im Beispiel verwendet wird. Sie müssen zuerst das Pixel Bender Toolkit verwenden, um den Quellcode in eine Pixel Bender-Bytecodedatei zu kompilieren. Gehen Sie bei der Erstellung der Pixel Bender-Bytecodeatei wie folgt vor:

1. Öffnen Sie das Adobe Pixel Bender Toolkit. Wählen Sie bei Bedarf aus dem Menü „Build“ (Erstellen) die Option „Turn on Flash Player warnings and errors“ (Flash Player-Warnungen und Fehlermeldungen aktivieren).
2. Kopieren Sie das Pixel Bender-Codebeispiel und fügen Sie sie im Codeeditorfenster im Pixel Bender Toolkit ein.
3. Wählen Sie aus dem Menü „File“ (Datei) die Option „Export kernel filter for Flash Player“ (Kernelfilter für Flash Player exportieren).
4. Speichern Sie die Pixel Bender-Bytecodedatei im gleichen Verzeichnis wie das Flash-Dokument. Der Dateiname sollte dem im Beispielverzeichnis angegebenen Namen entsprechen.

Der ActionScript-Teil jedes Beispiels wird als Klassendatei geschrieben. So testen Sie das Beispiel in Flash Professional:

1. Erstellen Sie ein leeres Flash-Dokument und speichern Sie es auf Ihrem Computer.
2. Erstellen Sie eine neue ActionScript-Datei und speichern Sie sie im selben Verzeichnis wie das Flash-Dokument. Der Name der Datei sollte mit dem Namen der Klasse im Codebeispiel übereinstimmen. Definiert die Codeliste zum Beispiel eine Klasse mit dem Namen „MyApplication“, speichern Sie die ActionScript-Datei unter dem Namen „MyApplication.as“.
3. Kopieren Sie das Codebeispiel in die ActionScript-Datei und speichern Sie die Datei.
4. Klicken Sie im Flash-Dokument auf eine leere Stelle der Bühne oder des Arbeitsbereichs, um den Eigenschafteninspektor des Dokuments zu aktivieren.
5. Geben Sie im Eigenschafteninspektor im Feld „Dokumentklasse“ den Namen der aus dem Text kopierten ActionScript-Klasse ein.
6. Starten Sie das Programm mit „Steuerung“ > „Film testen“.

   Die Ergebnisse des Codebeispiels werden im Vorschaufenster angezeigt.

Diese Techniken zum Testen von Beispielcode werden ausführlich unter „Verwendung der ActionScript-Beispiele“ auf Seite 1176 erläutert.

Laden oder Einbetten eines Shaders

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Um einen Pixel Bender-Shader in ActionScript zu verwenden, benötigen Sie zunächst Zugriff auf den Shader im ActionScript-Code. Da Shader mit dem Adobe Pixel Bender Toolkit erstellt und in der Pixel Bender-Programmiersprache geschrieben werden, ist kein direkter Zugriff in ActionScript möglich. Sie müssen stattdessen eine Instanz der Shader-Klasse erstellen, die den Pixel Bender-Shader in ActionScript repräsentiert. Mithilfe des Shader-Objekts können Sie Informationen über den Shader ermitteln, z. B. ob Parameter oder Werte für Eingabebilder erforderlich sind. Um den Shader zu verwenden, übergeben Sie das Shader-Objekt an andere Objekte. Um den Shader beispielsweise als Filter zu verwenden, weisen Sie das Shader-Objekt der `shader`-Eigenschaft eines ShaderFilter-Objekts zu. Alternativ können Sie den Shader auch als Füllung für eine Zeichnung verwenden, indem Sie das Shader-Objekt als Argument an die `Graphics.beginShaderFill()`-Methode übergeben.

Der ActionScript-Code kann auf einen mit dem Adobe Pixel Bender Toolkit erstellten Shader (eine .pbj-Datei) auf zwei Arten zugreifen:

- Laden zur Laufzeit: Die Shader-Datei kann mit einem URLLoader-Objekt als externes Element geladen werden. Dieses Verfahren entspricht dem Laden anderer externer Elemente wie z. B. einer Textdatei. Das folgende Beispiel zeigt, wie eine Bytecode-Datei für einen Shader zur Laufzeit geladen und mit einer Shader-Instanz verknüpft wird:

```
var loader:URLLoader = new URLLoader();
loader.dataFormat = URLLoaderDataFormat.BINARY;
loader.addEventListener(Event.COMPLETE, onLoadComplete);
loader.load(new URLRequest("myShader.pbj"));

var shader:Shader;

function onLoadComplete(event:Event):void {
    // Create a new shader and set the loaded data as its bytecode
    shader = new Shader();
    shader.byteCode = loader.data;

    // You can also pass the bytecode to the Shader() constructor like this:
    // shader = new Shader(loader.data);

     // do something with the shader
}
```

- Einbetten in die SWF-Datei: Die Shader-Datei kann beim Kompilieren mithilfe des Metadaten-Tags `[Embed]` in die SWF-Datei eingebettet werden. Das Metadaten-Tag `[Embed]` ist nur verfügbar, wenn Sie die SWF-Datei mit dem Flex SDK kompilieren. Der `source`-Parameter des `[Embed]`-Tags verweist auf die Shader-Datei und der `mimeType`-Parameter lautet `"application/octet-stream"`, wie im folgenden Beispiel gezeigt:

```
[Embed(source="myShader.pbj", mimeType="application/octet-stream")]
var MyShaderClass:Class;

// ...

// create a shader and set the embedded shader as its bytecode
var shader:Shader = new Shader();
shader.byteCode = new MyShaderClass();

// You can also pass the bytecode to the Shader() constructor like this:
// var shader:Shader = new Shader(new MyShaderClass());

// do something with the shader
```

In beiden Fällen verknüpfen Sie den unformatierten Shader-Bytecode (die `URLLoader.data`-Eigenschaft oder eine Instanz der `[Embed]`-Datenklasse) mit der Shader-Instanz. Wie im vorherigen Beispiel zu sehen ist, können Sie den Bytecode auf zwei Arten mit der Shader-Instanz verknüpfen. Sie können den Shader-Bytecode als Argument an den `Shader()`-Konstruktor übergeben. Oder Sie legen ihn als `byteCode`-Eigenschaft der Shader-Instanz fest.

Nachdem ein Pixel Bender-Shader erstellt und mit einem Shader-Objekt verknüpft wurde, können Sie mit dem Shader auf unterschiedliche Weise Effekte erstellen. Sie können den Shader als Filter, Mischmodus oder Bitmap-Füllung sowie zur eigenständigen Verarbeitung von Bitmaps und anderen Daten verwenden. Darüber hinaus können Sie mit der `data`-Eigenschaft des Shader-Objekts auf die Metadaten des Shaders zugreifen, Eingabebilder angeben und Parameterwerte festlegen.

Zugreifen auf Shader-Metadaten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Beim Erstellen eines Shader-Kernels in Pixel Bender kann der Ersteller im Pixel Bender-Quellcode Metadaten zum Shader angeben. Bei der Verwendung eines Shaders in ActionScript haben Sie die Möglichkeit, diesen zu analysieren und die zugehörigen Metadaten zu extrahieren.

Wenn Sie eine Shader-Instanz erstellen und mit einem Pixel Bender-Shader verknüpfen, wird ein ShaderData-Objekt mit Daten zu dem Shader erstellt und in der `data`-Eigenschaft des Shader-Objekts gespeichert. Die ShaderData-Klasse selbst definiert keine Eigenschaften. Zur Laufzeit wird dem ShaderData-Objekt jedoch dynamisch eine Eigenschaft für jeden Metadatenwert hinzugefügt, der im Quellcode des Shaders definiert ist. Der Name der einzelnen Eigenschaften entspricht dem Namen, der in den Metadaten angegeben ist. Angenommen, der Quellcode eines Pixel Bender-Shaders enthält die folgende Metadatendefinition:

```
namespace : "Adobe::Example";
vendor : "Bob Jones";
version : 1;
description : "Creates a version of the specified image with the specified brightness.";
```

Das ShaderData-Objekt für diesen Shader wird dann mit den folgenden Eigenschaften und Werten erstellt:

- `namespace` (String): `"Adobe::Example"`
- `vendor` (String): `"Bob Jones"`
- `version` (String): `"1"`
- `description` (String): `"Creates a version of the specified image with the specified brightness"`

Da Metadateneigenschaften dem ShaderData-Objekt dynamisch hinzugefügt werden, können Sie eine `for..in`-Schleife verwenden, um das ShaderData-Objekt zu analysieren. Mit diesem Verfahren können Sie ermitteln, ob der Shader Metadaten enthält und wie die Metadatenwerte lauten. Zusätzlich zu Metadateneigenschaften kann ein ShaderData-Objekt Eigenschaften aufweisen, die in dem Shader definierte Eingaben und Parameter darstellen. Bei Verwendung einer `for..in`-Schleife zum Analysieren eines ShaderData-Objekts müssen Sie den Datentyp jeder Eigenschaft überprüfen. Auf diese Weise können Sie feststellen, ob die Eigenschaft eine Eingabe (eine ShaderInput-Instanz), ein Parameter (eine ShaderParameter-Instanz) oder ein Metadatenwert (eine String-Instanz) ist. Das folgende Beispiel veranschaulicht, wie Sie mit der `for..in`-Schleife die dynamischen Eigenschaften der `data`-Eigenschaft eines Shaders überprüfen. Jeder Metadatenwert wird einer Vector-Instanz mit dem Namen `metadata` hinzugefügt. Beachten Sie, dass für dieses Beispiel bereits eine Shader-Instanz mit dem Namen `myShader` erstellt sein muss.

```
var shaderData:ShaderData = myShader.data;
var metadata:Vector.<String> = new Vector.<String>();

for (var prop:String in shaderData)
{
    if (!(shaderData[prop] is ShaderInput) && !(shaderData[prop] is ShaderParameter))
    {
        metadata[metadata.length] = shaderData[prop];
    }
}

// do something with the metadata
```

Eine Version dieses Beispiels, in der zudem Shader-Eingaben und -Parameter extrahiert werden, finden Sie unter „Identifizieren von Shader-Eingaben und -Parametern“ auf Seite 326. Weitere Informationen zu Eingabe- und Parametereigenschaften finden Sie unter „Angeben von Shader-Eingaben und -Parameterwerten“ auf Seite 326.

Angeben von Shader-Eingaben und -Parameterwerten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Definition zahlreicher Pixel Bender-Shader sieht vor, dass ein oder mehrere Eingabebilder für die Shader-Verarbeitung verwendet werden. Ein Shader kann beispielsweise ein Quellbild akzeptieren und dieses Bild mit einem bestimmten Effekt ausgeben, der darauf angewendet wurde. Abhängig von der Verwendung des Shaders kann der Eingabewert entweder automatisch festgelegt werden, oder Sie müssen einen Wert explizit angeben. Ebenso geben zahlreiche Shader Parameter an, mit deren Hilfe die Ausgabe des Shaders angepasst wird. Sie müssen für jeden Parameter explizit einen Wert festlegen, bevor Sie den Shader verwenden.

Verwenden Sie die `data`-Eigenschaft des Shader-Objekts, um Shader-Eingaben und -Parameter festzulegen und um zu bestimmen, ob ein bestimmter Shader Eingaben oder Parameter voraussetzt. Die `data`-Eigenschaft ist eine ShaderData-Instanz.

Identifizieren von Shader-Eingaben und -Parametern

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Beim Angeben von Shader-Eingaben und -Parameterwerten müssen Sie zunächst ermitteln, ob der verwendete Shader Eingabebilder oder Parameter voraussetzt. Jede Shader-Instanz verfügt über eine `data`-Eigenschaft, die ein ShaderData-Objekt enthält. Wenn der Shader Eingaben oder Parameter definiert, erfolgt der Zugriff auf diese als Eigenschaften des ShaderData-Objekts. Die Namen der Eigenschaften stimmen mit den Namen überein, die im Shader-Quellcode für die Eingaben und Parameter angegeben sind. Wenn der Shader beispielsweise eine Eingabe mit dem Namen `src` definiert, enthält das ShaderData-Objekt eine Eigenschaft mit dem Namen `src` für diese Eingabe. Jede Eigenschaft, die eine Eingabe darstellt, ist eine ShaderInput-Instanz. Eigenschaften, die einen Parameter darstellen, sind ShaderParameter-Instanzen.

Im Idealfall stellt der Ersteller des Shaders entsprechende Dokumentation bereit, in der angegeben wird, welche Eingabebildwerte und Parameter der Shader voraussetzt, wofür diese stehen, wie die zulässigen Werte lauten usw.

Wenn keine Dokumentation zum Shader verfügbar ist (und Sie den Quellcode nicht haben), können Sie die Shader-Daten analysieren, um die Eingaben und Parameter zu ermitteln. Die Eigenschaften, die Eingaben und Parameter repräsentieren, werden dynamisch zum ShaderData-Objekt hinzugefügt. Sie können daher eine `for..in`-Schleife nutzen, um das ShaderData-Objekt zu analysieren und zu ermitteln, ob der zugehörige Shader Eingaben oder Parameter definiert. Wie bereits im Abschnitt „Zugreifen auf Shader-Metadaten“ auf Seite 325 erläutert wurde, wird auf alle Metadatenwerte, die für einen Shader definiert sind, auch als dynamische Eigenschaft zugegriffen, die der `Shader.data`-Eigenschaft hinzugefügt wird. Wenn Sie dieses Verfahren zum Identifizieren von Shader-Eingaben und -Parametern verwenden, müssen Sie den Datentyp der dynamischen Eigenschaften überprüfen. Wenn eine Eigenschaft eine ShaderInput-Instanz ist, stellt diese eine Eingabe dar. Handelt es sich um eine ShaderParameter-Instanz, stellt sie einen Parameter dar. Andernfalls handelt es sich um einen Metadatenwert. Das folgende Beispiel veranschaulicht, wie Sie mit der `for..in`-Schleife die dynamischen Eigenschaften der `data`-Eigenschaft eines Shaders überprüfen. Jede Eingabe (ShaderInput-Objekt) wird einer Vector-Instanz mit dem Namen `inputs` hinzugefügt. Jeder Parameter (ShaderParameter-Objekt) wird einer Vector-Instanz mit dem Namen `parameters` hinzugefügt. Schließlich werden alle Metadateneigenschaften einer Vector-Instanz mit dem Namen `metadata` hinzugefügt. Beachten Sie, dass für dieses Beispiel bereits eine Shader-Instanz mit dem Namen `myShader` erstellt sein muss.

```
var shaderData:ShaderData = myShader.data;
var inputs:Vector.<ShaderInput> = new Vector.<ShaderInput>();
var parameters:Vector.<ShaderParameter> = new Vector.<ShaderParameter>();
var metadata:Vector.<String> = new Vector.<String>();

for (var prop:String in shaderData)
{
    if (shaderData[prop] is ShaderInput)
    {
        inputs[inputs.length] = shaderData[prop];
    }
    else if (shaderData[prop] is ShaderParameter)
    {
        parameters[parameters.length] = shaderData[prop];
    }
    else
    {
        metadata[metadata.length] = shaderData[prop];
    }
}

// do something with the inputs or properties
```

Angeben von Shader-Eingabewerten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Viele Shader setzen mindestens ein Eingabebild voraus, das in der Shader-Verarbeitung verwendet wird. Häufig wird eine Eingabe bei der Verwendung des Shader-Objekts automatisch angegeben. Beispiel: Ein Shader setzt eine Eingabe voraus und dieser Shader wird als Filter verwendet. Wenn der Filter auf ein Anzeigeobjekt oder ein BitmapData-Objekt angewendet wird, wird dieses Objekt automatisch als Eingabe festgelegt. In diesem Fall müssen Sie nicht explizit einen Eingabewert angeben.

In anderen Fällen, insbesondere wenn der Shader mehrere Eingaben definiert, legen Sie den Wert für eine Eingabe explizit fest. Jede Eingabe, die in einem Shader definiert ist, wird in ActionScript durch ein ShaderInput-Objekt dargestellt. Das ShaderInput-Objekt ist eine Eigenschaft der ShaderData-Instanz in der `data`-Eigenschaft des Shader-Objekts, wie unter „Identifizieren von Shader-Eingaben und -Parametern“ auf Seite 326 beschrieben. Beispiel: Ein Shader definiert eine Eingabe mit dem Namen `src` und dieser Shader ist mit einem Shader-Objekt mit dem Namen `myShader` verknüpft. In diesem Fall greifen Sie auf das ShaderInput-Objekt für die Eingabe `src` unter Verwendung des folgenden Bezeichners zu:

```
myShader.data.src
```

Jedes ShaderInput-Objekt verfügt über eine `input`-Eigenschaft, mit der der Wert für die Eingabe festgelegt wird. Legen Sie für die `input`-Eigenschaft eine BitmapData-Instanz fest, um Bilddaten anzugeben. Sie können für die `input`-Eigenschaft auch eine BitmapData- oder Vector.<Number>-Instanz festlegen, um binäre Daten oder Zahlen anzugeben. Weitere Informationen zur Verwendung einer BitmapData- oder Vector.<Number>-Instanz als Eingabe finden Sie im Eintrag zu `ShaderInput.input` im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Zusätzlich zur `input`-Eigenschaft verfügt ein ShaderInput-Objekt über Eigenschaften, mit denen Sie den Bildtyp ermitteln können, der von der Eingabe vorausgesetzt wird. Hierzu zählen die Eigenschaften `width`, `height` und `channels`. Jedes ShaderInput-Objekt verfügt außerdem über eine `index`-Eigenschaft, anhand derer Sie feststellen können, ob für die Eingabe ein expliziter Wert angegeben werden muss. Wenn ein Shader mehr Eingaben als die automatisch festgelegte Anzahl erfordert, müssen Sie Werte für diese Eingaben festlegen. Weitere Informationen zu den Verwendungsmöglichkeiten von Shadern sowie zur automatischen Einstellung von Eingabewerten finden Sie unter „Verwenden eines Shaders“ auf Seite 331.

Angeben von Shader-Parameterwerten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Einige Shader definieren Parameterwerte, die der Shader zum Erstellen der Ergebnisse nutzt. Ein Shader, der die Helligkeit eines Bilds ändert, kann beispielsweise einen Parameter für die Helligkeit angeben, der festlegt, inwieweit die Helligkeit durch den Vorgang angepasst wird. Für einen einzelnen, in einem Shader definierten Parameter können ein oder mehrere Werte vorausgesetzt werden, je nach Parameterdefinition im Shader. Jeder Parameter, der in einem Shader definiert ist, wird in ActionScript durch ein ShaderParameter-Objekt dargestellt. Das ShaderParameter-Objekt ist eine Eigenschaft der ShaderData-Instanz in der data-Eigenschaft des Shader-Objekts, wie unter „Identifizieren von Shader-Eingaben und -Parametern“ auf Seite 326 beschrieben. Beispiel: Ein Shader definiert einen Parameter mit dem Namen `brightness` und dieser Shader wird durch ein Shader-Objekt mit dem Namen `myShader` dargestellt. In diesem Fall greifen Sie auf das ShaderParameter-Objekt für den Parameter `brightness` unter Verwendung des folgenden Bezeichners zu:

```
myShader.data.brightness
```

Um einen oder mehrere Werte für den Parameter anzugeben, erstellen Sie ein ActionScript-Array mit den Werten und weisen dieses Array der `value`-Eigenschaft des ShaderParameter-Objekts zu. Die `value`-Eigenschaft ist als Array-Instanz definiert, da ein einzelner Shader-Parameter mehrere Werte voraussetzen kann. Selbst wenn der Shader-Parameter nur einen Wert erfordert, müssen Sie den Wert in das Array-Objekt einschließen, um ihn der `ShaderParameter.value`-Eigenschaft zuzuweisen. Im Folgenden wird die Einstellung eines einzelnen Werts als `value`-Eigenschaft erläutert.

```
myShader.data.brightness.value = [75];
```

Wenn der Pixel Bender-Quellcode des Shaders einen Standardwert für den Parameter definiert, wird beim Erstellen des Shader-Objekts ein Array mit den Standardwerten erstellt und der `value`-Eigenschaft des ShaderParameter-Objekts zugewiesen. Nachdem der `value`-Eigenschaft ein Array zugewiesen wurde (auch wenn es das Standard-Array ist), können Sie den Parameterwert ändern, indem Sie den Wert des Array-Elements bearbeiten. Sie müssen kein neues Array erstellen und der `value`-Eigenschaft zuweisen.

Das folgende Beispiel veranschaulicht, wie Sie den Parameterwert eines Shaders in ActionScript festlegen. In diesem Beispiel definiert der Shader einen Parameter mit dem Namen `color`. Der `color`-Parameter ist als eine `float4`-Variable im Pixel Bender-Quellcode deklariert, d. h. es handelt sich um ein Array mit vier Gleitkommazahlen. In diesem Beispiel wird der Wert des `color`-Parameters kontinuierlich geändert. Bei jeder Änderung wird der Shader verwendet, um ein farbiges Rechteck auf dem Bildschirm zu zeichnen. Das Ergebnis ist eine animierte Farbänderung.

Hinweis: *Der Code für dieses Beispiel wurde von Ryan Taylor geschrieben. Vielen Dank an Ryan für die Bereitstellung dieses Beispiels. Unter www.boostworthy.com/ können Sie Ryans Mappe einsehen und seine Texte lesen.*

Der ActionScript-Code basiert primär auf drei Methoden:

- `init()`: In der `init()`-Methode lädt der Code die Pixel Bender-Bytecodedatei, die den Shader enthält. Beim Laden der Datei wird die `onLoadComplete()`-Methode aufgerufen.

- `onLoadComplete()`: In der `onLoadComplete()`-Methode erstellt der Code das Shader-Objekt mit dem Namen `shader`. Er erstellt außerdem eine Sprite-Instanz mit dem Namen `texture`. In der `renderShader()`-Methode zeichnet der Code das Shader-Ergebnis pro Bild einmal in `texture`.
- `onEnterFrame()`: Die `onEnterFrame()`-Methode wird einmal pro Bild aufgerufen und erzeugt den Animationseffekt. In dieser Methode legt der Code für den Shader-Parameterwert die neue Farbe fest. Anschließend wird die `renderShader()`-Methode aufgerufen, um das Ergebnis des Shaders in Form eines Rechtecks zu zeichnen.
- `renderShader()`: In der `renderShader()`-Methode ruft der Code die `Graphics.beginShaderFill()`-Methode auf, um eine Füllung für den Shader anzugeben. Anschließend wird ein Rechteck gezeichnet, dessen Füllung durch die Shader-Ausgabe (die generierte Farbe) definiert wird. Weitere Informationen zu dieser Verwendung des Shaders finden Sie unter „Verwenden eines Shaders als Füllmuster für Zeichnungen“ auf Seite 332.

Im Folgenden wird der ActionScript-Code für dieses Beispiel angezeigt. Verwenden Sie diese Klasse als Hauptanwendungsklasse für ausschließlich in ActionScript erstellte Projekte in Flash Builder oder als Dokumentklasse für die FLA-Datei in Flash Professional:

```
package
{
    import flash.display.Shader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class ColorFilterExample extends Sprite
    {
        private const DELTA_OFFSET:Number = Math.PI * 0.5;
        private var loader:URLLoader;
        private var shader:Shader;
        private var texture:Sprite;
        private var delta:Number = 0;

        public function ColorFilterExample()
        {
            init();
        }

        private function init():void
        {
            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("ColorFilter.pbj"));
        }

        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);

            texture = new Sprite();

            addChild(texture);

            addEventListener(Event.ENTER_FRAME, onEnterFrame);
```

```
        }
        private function onEnterFrame(event:Event):void
        {
            shader.data.color.value[0] = 0.5 + Math.cos(delta - DELTA_OFFSET) * 0.5;
            shader.data.color.value[1] = 0.5 + Math.cos(delta) * 0.5;
            shader.data.color.value[2] = 0.5 + Math.cos(delta + DELTA_OFFSET) * 0.5;
            // The alpha channel value (index 3) is set to 1 by the kernel's default
            // value. This value doesn't need to change.

            delta += 0.1;

            renderShader();
        }

        private function renderShader():void
        {
            texture:graphics.clear();
            texture.graphics.beginShaderFill(shader);
            texture.graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
            texture.graphics.endFill();
        }
    }
}
```

Im Folgenden ist der Quellcode für den Kernel des ColorFilter-Shaders dargestellt, der zum Erstellen der Pixel Bender-Bytecodedatei „ColorFilter.pbj“ verwendet wurde:

```
<languageVersion : 1.0;>
kernel ColorFilter
<
    namespace : "boostworthy::Example";
    vendor : "Ryan Taylor";
    version : 1;
    description : "Creates an image where every pixel has the specified color value.";
>
{
    output pixel4 result;

    parameter float4 color
    <
        minValue:float4(0, 0, 0, 0);
        maxValue:float4(1, 1, 1, 1);
        defaultValue:float4(0, 0, 0, 1);
    >;

    void evaluatePixel()
    {
        result = color;
    }
}
```

Wenn Sie einen Shader verwenden, dessen Parameter nicht dokumentiert sind, können Sie die Art und Anzahl der Elemente, die in das Array eingeschlossen werden müssen, durch Überprüfen der `type`-Eigenschaft des ShaderParameter-Objekts ermitteln. Die `type`-Eigenschaft gibt den Datentyp des Parameters an, der im Shader selbst definiert ist. Eine Liste mit Informationen zu Art und Anzahl der Elemente, die von jedem Parametertyp vorausgesetzt werden, finden Sie im ActionScript 3.0-Referenzhandbuch unter `ShaderParameter.value`.

Jedes ShaderParameter-Objekt verfügt zudem über eine `index`-Eigenschaft, die die Position des Parameters in der Reihenfolge der Shader-Parameter angibt. Neben diesen Eigenschaften kann ein ShaderParameter-Objekt weitere Eigenschaften aufweisen, die vom Ersteller des Shaders bereitgestellte Metadatenwerte enthalten. Der Ersteller kann beispielsweise Metadatenwerte wie Minimum und Maximum sowie Standardwerte für einen Parameter festlegen. Alle vom Ersteller angegebenen Metadatenwerte werden dem ShaderParameter-Objekt als dynamische Eigenschaften hinzugefügt. Verwenden Sie zum Analysieren dieser Eigenschaften eine `for..in`-Schleife, um die dynamischen Eigenschaften des ShaderParameter-Objekts anzuzeigen und die zugehörigen Metadaten zu identifizieren. Das folgende Beispiel veranschaulicht, wie Sie mit der `for..in`-Schleife die Metadaten eines ShaderParameter-Objekts identifizieren. Jeder Metadatenwert wird einer Vector-Instanz mit dem Namen `metadata` hinzugefügt. Beachten Sie, dass für dieses Beispiel eine Shader-Instanz mit dem Namen `myShader` bereits erstellt sein muss und dass diese einen Parameter mit dem Namen `brightness` aufweisen muss:

```
var brightness:ShaderParameter = myShader.data.brightness;
var metadata:Vector.<String> = new Vector.<String>();

for (var prop:String in brightness)
{
    if (brightness[prop] is String)
    {
        metadata[metadata.length] = brightness[prop];
    }
}

// do something with the metadata
```

Verwenden eines Shaders

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sobald ein Pixel Bender-Shader als Shader-Objekt in ActionScript verfügbar ist, kann er auf unterschiedliche Weise verwendet werden:

- Shader als Füllmuster für Zeichnungen: Der Shader definiert den Füllbereich einer mit der Zeichnungs-API gezeichneten Form.
- Füllmodus: Der Shader definiert den Übergang zwischen zwei einander überlappenden Anzeigeobjekten.
- Filter: Der Shader definiert einen Filter, der das Erscheinungsbild der visuellen Inhalte modifiziert.
- Eigenständige Shader-Verarbeitung: Die Verarbeitung des Shaders wird ausgeführt, ohne die beabsichtigte Verwendung der Ausgabe anzugeben. Der Shader kann optional im Hintergrund ausgeführt werden. Das Ergebnis ist dann nach Abschluss der Verarbeitung verfügbar. Mit diesem Verfahren können Bitmap-Daten erzeugt und nicht-grafische Daten verarbeitet werden.

Verwenden eines Shaders als Füllmuster für Zeichnungen

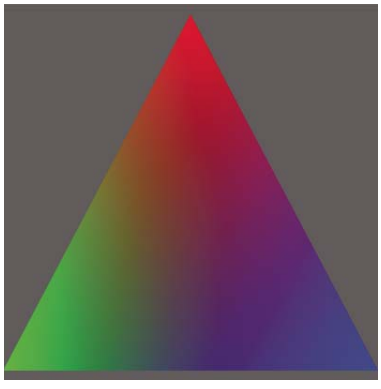
Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wenn Sie eine Zeichnungsfüllung mit einem Shader erstellen, verwenden Sie die Zeichnungs-API-Methoden, um eine Vektorform zu erzeugen. Die Ausgabe des Shaders wird verwendet, um die Form zu füllen, so wie jedes Bitmap-Bild mit der Zeichnungs-API als Bitmap-Füllung verwendet werden kann. Um eine Shader-Füllung zu erzeugen, rufen Sie an dem Punkt in Ihrem Code, an dem Sie mit dem Zeichnen der Form beginnen möchten, die `beginShaderFill()`-Methode des Graphics-Objekts auf. Übergeben Sie das Shader-Objekt wie in diesem Beispiel dargestellt als erstes Argument an die `beginShaderFill()`-Methode:

```
var canvas:Sprite = new Sprite();
canvas.graphics.beginShaderFill(myShader);
canvas.graphics.drawRect(10, 10, 150, 150);
canvas.graphics.endFill();
// add canvas to the display list to see the result
```

Wenn Sie einen Shader als Zeichnungsfüllung verwenden, können Sie die vom Shader benötigten Eingabebild- und Parameterwerte beliebig setzen.

Im folgenden Beispiel wird gezeigt, wie ein Shader als Zeichnungsfüllung verwendet wird. In diesem Beispiel erstellt der Shader einen Drei-Punkte-Verlauf. Dieser Verlauf enthält drei Farben, jede an der Spitze eines Dreiecks, die zur Mitte hin ineinander übergehen. Zusätzlich drehen sich die Farben, um einen horizontalen Farbdreheffekt zu erzeugen.



Hinweis: *Der Code für dieses Beispiel wurde von Petri Leskinen geschrieben. Vielen Dank an Petri für die Bereitstellung dieses Beispiels. Weitere Beispiele und Hilfestellungen von Petri können Sie unter http://pixelero.wordpress.com/ einsehen.*

Der ActionScript-Code befindet sich in drei Methoden:

- `init()`: Die `init()`-Methode wird beim Laden der Anwendung aufgerufen. In dieser Methode setzt der Code die Anfangswerte für die Point-Objekte, die die Spitzen des Dreiecks bilden. Zudem erstellt der Code eine Sprite-Instanz mit dem Namen `canvas`. Später zeichnet der Code in `updateShaderFill()` das Shader-Ergebnis einmal pro Bild in `canvas`. Schließlich lädt der Code die Shader-Bytecodedatei.
- `onLoadComplete()`: In der `onLoadComplete()`-Methode erstellt der Code das Shader-Objekt mit dem Namen `shader`. Er setzt außerdem die Anfangswerte der Parameter. Schließlich fügt der Code die `updateShaderFill()`-Methode als Listener für das Ereignis `enterFrame` hinzu, sie wird also pro Bild einmal aufgerufen, um einen Animationseffekt zu erzielen.

- `updateShaderFill()`: Die `updateShaderFill()`-Methode wird pro Bild einmal aufgerufen und erzeugt so einen Animationseffekt. In dieser Methode berechnet und setzt der Code die Parameterwerte des Shaders. Der Code ruft dann die `beginShaderFill()`-Methode auf, um eine Shader-Füllung zu erzeugen, und ruft andere Zeichnungs-API-Methoden auf, um das Shader-Ergebnis in ein Dreieck zu zeichnen.

Im Folgenden wird der ActionScript-Code für dieses Beispiel angezeigt. Verwenden Sie diese Klasse als Hauptanwendungsklasse für ausschließlich in ActionScript erstellte Projekte in Flash Builder oder als Dokumentklasse für eine FLA-Datei in Flash Professional:

```
package
{
    import flash.display.Shader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.geom.Point;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class ThreePointGradient extends Sprite
    {
        private var canvas:Sprite;
        private var shader:Shader;
        private var loader:URLLoader;

        private var topMiddle:Point;
        private var bottomLeft:Point;
        private var bottomRight:Point;

        private var colorAngle:Number = 0.0;
        private const d120:Number = 120 / 180 * Math.PI; // 120 degrees in radians


        public function ThreePointGradient()
        {
            init();
        }

        private function init():void
        {
            canvas = new Sprite();
            addChild(canvas);

            var size:int = 400;
            topMiddle = new Point(size / 2, 10);
            bottomLeft = new Point(0, size - 10);
            bottomRight = new Point(size, size - 10);

            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("ThreePointGradient.pbj"));
        }

        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);
```

```
            shader.data.point1.value = [topMiddle.x, topMiddle.y];
            shader.data.point2.value = [bottomLeft.x, bottomLeft.y];
            shader.data.point3.value = [bottomRight.x, bottomRight.y];

            addEventListener(Event.ENTER_FRAME, updateShaderFill);
        }

        private function updateShaderFill(event:Event):void
        {
            colorAngle += .06;

            var c1:Number = 1 / 3 + 2 / 3 * Math.cos(colorAngle);
            var c2:Number = 1 / 3 + 2 / 3 * Math.cos(colorAngle + d120);
            var c3:Number = 1 / 3 + 2 / 3 * Math.cos(colorAngle - d120);

            shader.data.color1.value = [c1, c2, c3, 1.0];
            shader.data.color2.value = [c3, c1, c2, 1.0];
            shader.data.color3.value = [c2, c3, c1, 1.0];

            canvas.graphics.clear();
            canvas.graphics.beginShaderFill(shader);

            canvas.graphics.moveTo(topMiddle.x, topMiddle.y);
            canvas.graphics.lineTo(bottomLeft.x, bottomLeft.y);
            canvas.graphics.lineTo(bottomRight.x, bottomLeft.y);

            canvas.graphics.endFill();
        }
    }
}
```

Im Folgenden ist der Quellcode für den Kernel des ThreePointGradient-Shaders dargestellt, der zum Erstellen der Pixel Bender-Bytecodedatei „ThreePointGradient.pbj“ verwendet wurde:

```
<languageVersion : 1.0;>
kernel ThreePointGradient
<
    namespace : "Petri Leskinen::Example";
    vendor : "Petri Leskinen";
    version : 1;
    description : "Creates a gradient fill using three specified points and colors.";
>
{
    parameter float2 point1 // coordinates of the first point
    <
        minValue:float2(0, 0);
        maxValue:float2(4000, 4000);
        defaultValue:float2(0, 0);
    >;

    parameter float4 color1 // color at the first point, opaque red by default
    <
        defaultValue:float4(1.0, 0.0, 0.0, 1.0);
    >;

    parameter float2 point2 // coordinates of the second point
```

```
    <
        minValue:float2(0, 0);
        maxValue:float2(4000, 4000);
        defaultValue:float2(0, 500);
    >;

    parameter float4 color2 // color at the second point, opaque green by default
    <
        defaultValue:float4(0.0, 1.0, 0.0, 1.0);
    >;

    parameter float2 point3 // coordinates of the third point
    <
        minValue:float2(0, 0);
        maxValue:float2(4000, 4000);
        defaultValue:float2(0, 500);
    >;

    parameter float4 color3 // color at the third point, opaque blue by default
    <
        defaultValue:float4(0.0, 0.0, 1.0, 1.0);
    >;

    output pixel4 dst;

    void evaluatePixel()
    {
        float2 d2 = point2 - point1;
        float2 d3 = point3 - point1;

        // transformation to a new coordinate system
        // transforms point 1 to origin, point2 to (1, 0), and point3 to (0, 1)
        float2x2 mtrx = float2x2(d3.y, -d2.y, -d3.x, d2.x) / (d2.x * d3.y - d3.x * d2.y);
        float2 pNew = mtrx * (outCoord() - point1);

        // repeat the edge colors on the outside
        pNew.xy = clamp(pNew.xy, 0.0, 1.0); // set the range to 0.0 ... 1.0

        // interpolating the output color or alpha value
        dst = mix(mix(color1, color2, pNew.x), color3, pNew.y);
    }
}
```

Hinweis: *Wenn Sie beim Rendern mit der GPU eine Shaderfüllung verwenden, hat der gefüllte Bereich die Farbe Cyan.*

Weitere Informationen zum Zeichnen von Formen mit der Zeichnungs-API finden Sie unter „Verwenden der Zeichnungs-API“ auf Seite 236.

Verwenden eines Shaders als Mischmodus

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Verwendung eines Shaders als Mischmodus ist nicht anders als die Verwendung anderer Mischmodi. Der Shader definiert das Erscheinungsbild, das durch das visuelle Mischen zweier Anzeigeobjekte entsteht. Um einen Shader als Mischmodus zu verwenden, weisen Sie Ihr Shader-Objekt der `blendShader`-Eigenschaft des Vordergrundanzeigeobjekts zu. Die Zuweisung eines anderen Werts als `null` zur `blendShader`-Eigenschaft setzt die `blendMode`-Eigenschaft des Anzeigeobjekts automatisch auf `BlendMode.SHADER`. Im folgenden Beispiel wird die Verwendung eines Shaders als Mischmodus verdeutlicht. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass ein Anzeigeobjekt mit dem Namen `foreground` im gleichen übergeordneten Objekt auf der Anzeigeliste wie andere Anzeigeinhalte enthalten ist, wobei `foreground` andere Inhalte überlappt.

```
foreground.blendShader = myShader;
```

Wenn Sie einen Shader als Mischmodus verwenden, muss der Shader mit mindestens zwei Eingaben definiert werden. Wie im Beispiel dargestellt, setzen Sie die Eingabewerte nicht in Ihrem Code. Stattdessen werden die beiden gemischten Bilder automatisch als Shader-Eingaben verwendet. Das Vordergrundbild wurde als zweites Bild gesetzt. (Dies ist das Anzeigeobjekt, auf das der Mischmodus angewendet wird.) Ein Hintergrundbild wird durch die Zusammenfügung aller Pixel hinter dem Begrenzungsrahmen des Vordergrundbilds erzeugt. Dieses Hintergrundbild wird als erstes Eingabebild gesetzt. Wenn Sie einen Shader verwenden, der mehr als zwei Eingaben erwartet, geben Sie einen Wert für die Eingaben an, die über die ersten beiden hinausgehen.

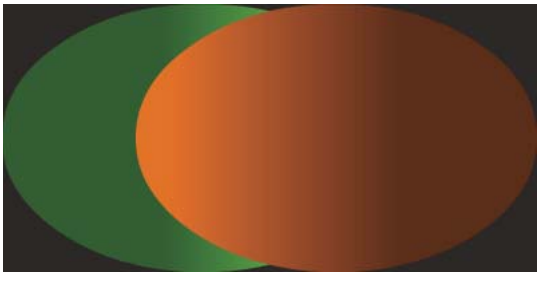
Im folgenden Beispiel wird die Verwendung eines Shaders als Mischmodus verdeutlicht. In diesem Beispiel wird ein aufhellender Mischmodus basierend auf Luminanz verwendet. Im Ergebnis der Mischung wird der hellste Pixelwert der miteinander gemischten Objekte angezeigt.

Hinweis: *Der Code für dieses Beispiel wurde von Mario Klingemann geschrieben. Vielen Dank an Mario für die Bereitstellung dieses Beispiels. Sie können Marios Arbeiten und Texte unter www.quasimondo.com/ einsehen.*

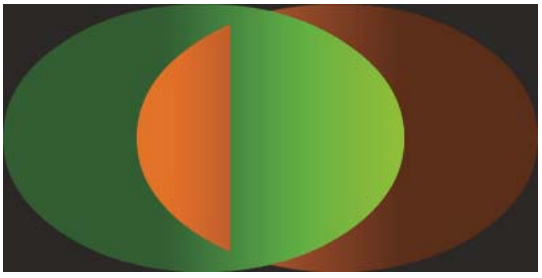
Der wichtige ActionScript-Code befindet sich in diesen beiden Methoden:

- `init()`: Die `init()`-Methode wird beim Laden der Anwendung aufgerufen. In dieser Methode lädt der Code die Shader-Bytecodedatei.
- `onLoadComplete()`: In der `onLoadComplete()`-Methode erstellt der Code das Shader-Objekt mit dem Namen `shader`. Der Code zeichnet dann drei Objekte: Das erste, `backdrop`, ist ein dunkelgrauer Hintergrund hinter den eingemischten Objekten. Das zweite, `backgroundShape`, ist eine grün verlaufende Ellipse. Das dritte Objekt, `foregroundShape`, ist eine in Orange verlaufende Ellipse.

  Die `foregroundShape`-Ellipse ist das Vordergrundobjekt der Mischung. Das Hintergrundbild der Mischung wird durch den Teil von `backdrop` und `backgroundShape` erzeugt, der durch den Begrenzungsrahmen des `foregroundShape`-Objekts überlappt wird. Das `foregroundShape`-Objekt ist das erste Objekt in der Anzeigeliste. Es überlappt das `backgroundShape`-Objekt teilweise und das `backdrop`-Objekt vollständig. Aufgrund dieser Überlappung würde die orangefarbene Ellipse (`foregroundShape`) ohne Mischmodus vollständig angezeigt und die grüne Ellipse (`backgroundShape`) würde teilweise von ihr überdeckt:



Mit aktiviertem Mischmodus scheint der hellere Teil der grünen Ellipse jedoch durch, da er heller ist als der Teil des `foregroundShape`-Objekts, das sie überlappt:



Im Folgenden wird der ActionScript-Code für dieses Beispiel angezeigt. Verwenden Sie diese Klasse als Hauptanwendungsklasse für ausschließlich in ActionScript erstellte Projekte in Flash Builder oder als Dokumentklasse für die FLA-Datei in Flash Professional:

```
package
{
    import flash.display.BlendMode;
    import flash.display.GradientType;
    import flash.display.Graphics;
    import flash.display.Shader;
    import flash.display.Shape;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.geom.Matrix;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class LumaLighten extends Sprite
    {
        private var shader:Shader;
        private var loader:URLLoader;

        public function LumaLighten()
        {
            init();
        }

        private function init():void
        {
            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("LumaLighten.pbj"));
        }


        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);

            var backdrop:Shape = new Shape();
            var g0:Graphics = backdrop.graphics;
```

```
            g0.beginFill(0x303030);
            g0.drawRect(0, 0, 400, 200);
            g0.endFill();
            addChild(backdrop);

            var backgroundShape:Shape = new Shape();
            var g1:Graphics = backgroundShape.graphics;
            var c1:Array = [0x336600, 0x80ff00];
            var a1:Array = [255, 255];
            var r1:Array = [100, 255];
            var m1:Matrix = new Matrix();
            m1.createGradientBox(300, 200);
            g1.beginGradientFill(GradientType.LINEAR, c1, a1, r1, m1);
            g1.drawEllipse(0, 0, 300, 200);
            g1.endFill();
            addChild(backgroundShape);

            var foregroundShape:Shape = new Shape();
            var g2:Graphics = foregroundShape.graphics;
            var c2:Array = [0xff8000, 0x663300];
            var a2:Array = [255, 255];
            var r2:Array = [100, 255];
            var m2:Matrix = new Matrix();
            m2.createGradientBox(300, 200);
            g2.beginGradientFill(GradientType.LINEAR, c2, a2, r2, m2);
            g2.drawEllipse(100, 0, 300, 200);
            g2.endFill();
            addChild(foregroundShape);

            foregroundShape.blendShader = shader;
            foregroundShape.blendMode = BlendMode.SHADER;
        }
    }
}
```

Im Folgenden ist der Quellcode für den Kernel des LumaLighten-Shaders dargestellt, der zum Erstellen der Pixel Bender-Bytecodedatei „LumaLighten.pbj“ verwendet wurde:

```
<languageVersion : 1.0;>
kernel LumaLighten
<
    namespace : "com.quasimondo.blendModes";
    vendor : "Quasimondo.com";
    version : 1;
    description : "Luminance based lighten blend mode";
>
{
    input image4 background;
    input image4 foreground;

    output pixel4 dst;

    const float3 LUMA = float3(0.212671, 0.715160, 0.072169);

    void evaluatePixel()
    {
        float4 a = sampleNearest(foreground, outCoord());
        float4 b = sampleNearest(background, outCoord());
        float luma_a = a.r * LUMA.r + a.g * LUMA.g + a.b * LUMA.b;
        float luma_b = b.r * LUMA.r + b.g * LUMA.g + b.b * LUMA.b;

        dst = luma_a > luma_b ? a : b;
    }
}
```

Weitere Informationen zur Verwendung von Mischmodi finden Sie unter „Anwenden von Mischmodi“ auf Seite 198.

Hinweis: *Wenn ein Pixel Bender-Shaderprogramm im Mischmodus in Flash Player oder AIR ausgeführt wird, verhalten sich die Sampling- und die `outCoord()`-Funktionen anders als in jedem anderen Kontext. Im Mischmodus gibt die Sampling-Funktion immer das aktuelle Pixel zurück, das vom Shader evaluiert wird. Beispielsweise können Sie kein Offset in Bezug auf `outCoord()` verwenden, um das Sampling eines angrenzenden Pixels durchzuführen. Genauso werden bei Verwendung der `outCoord()`-Funktion außerhalb einer Sampling-Funktion die Koordinaten immer mit 0 ausgewertet. Es ist beispielsweise nicht möglich, mit der Position eines Pixels zu beeinflussen, wie die gemischten Bilder kombiniert werden.*

Verwenden eines Shaders als Filter

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Verwendung eines Shaders als Filters unterscheidet sich nicht von der Verwendung anderer Filter in ActionScript. Wenn Sie einen Shader als Filter verwenden, wird das gefilterte Bild (ein Anzeigeobjekt oder BitmapData-Objekt) an den Shader übergeben. Der Shader verwendet das Eingabebild zur Erstellung der Filterausgabe, die gewöhnlich eine veränderte Version des Originalbildes ist. Handelt es sich bei dem gefilterten Objekt um ein Anzeigeobjekt, wird die Ausgabe des Shaders anstelle des gefilterten Anzeigeobjekts auf dem Bildschirm angezeigt. Handelt es sich bei dem gefilterten Objekt um ein BitmapData-Objekt, wird die Ausgabe des Shaders zum Inhalt des BitmapData-Objekts, dessen `applyFilter()`-Methode aufgerufen wird.

Um einen Shader als Filter zu verwenden, müssen Sie zunächst wie unter „Laden oder Einbetten eines Shaders“ auf Seite 323 beschrieben ein Shader-Objekt erstellen. Dann erstellen Sie ein mit dem Shader-Objekt verknüpftes ShaderFilter-Objekt. Das ShaderFilter-Objekt ist der Filter, der auf das gefilterte Objekt angewendet wird. Sie wenden das Objekt wie andere Filter auch auf Objekte an. Sie übergeben es an die `filters`-Eigenschaft eines Anzeigeobjekts oder Sie rufen die `applyFilter()`-Methode für ein BitmapData-Objekt auf. Mit dem folgenden Code wird zum Beispiel ein ShaderFilter-Objekt erstellt und der Filter auf ein Anzeigeobjekt mit dem Namen `homeButton` angewendet.

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
homeButton.filters = [myFilter];
```

Wenn Sie einen Shader als Filter verwenden, muss der Shader mit mindestens einer Eingabe definiert werden. Wie im Beispiel dargestellt, setzen Sie den Eingabewert nicht in Ihrem Code. Stattdessen wird das gefilterte Objekt oder BitmapData-Objekt als Eingabebild gesetzt. Wenn Sie einen Shader verwenden, der mehr als eine Eingabe erwartet, geben Sie einen Wert für die Eingaben an, die über die erste hinausgehen.

In einigen Fällen verändert ein Filter die Maße eines Originalbilds. Ein typischer Schlagschatteneffekt fügt zum Beispiel zusätzliche Pixel hinzu, die den Schatten enthalten, der zum Bild hinzugefügt wird. Wenn Sie einen Shader verwenden, der die Bildmaße ändert, setzen Sie die Eigenschaften `leftExtension`, `rightExtension`, `topExtension` und `bottomExtension`, um anzugeben, um wie viel die Bildmaße geändert werden sollen.

Im folgenden Beispiel wird gezeigt, wie ein Shader als Filter verwendet wird. Der Filter in diesem Beispiel kehrt die roten, grünen und blauen Kanalwerte eines Bildes um. Das Ergebnis ist ein „Negativ“ des Bildes.

Hinweis: *Der Shader in diesem Beispiel verwendet den invertRGB.pbk-Pixel Bender-Kernel, der im Pixel Bender Toolkit enthalten ist. Sie können den Quellcode für den Kernel aus dem Installationsverzeichnis des Pixel Bender Toolkits laden. Kompilieren Sie den Quellcode und speichern Sie die Bytecodedatei im gleichen Verzeichnis wie den Quellcode.*

Der wichtige ActionScript-Code befindet sich in diesen beiden Methoden:

- `init()`: Die `init()`-Methode wird beim Laden der Anwendung aufgerufen. In dieser Methode lädt der Code die Shader-Bytecodedatei.
- `onLoadComplete()`: In der `onLoadComplete()`-Methode erstellt der Code das Shader-Objekt mit dem Namen `shader`. Er erstellt und zeichnet dann den Inhalt eines Objekts mit dem Namen `target`. Das `target`-Objekt ist ein mit einer linear verlaufenden Farbe gefülltes Rechteck, links beginnend mit Rot über Grün und abschließend mit Hellblau. Das ungefilterte Objekt sieht wie folgt aus:

Nach Anwendung des Filters sind die Farben umgekehrt und das Rechteck sieht wie folgt aus:

Der Shader in diesem Beispiel verwendet den invertRGB.pbk-Pixel Bender-Kernel, der im Pixel Bender Toolkit enthalten ist. Der Quellcode steht in der Datei „invertRGB.pbk“ im Installationsverzeichnis des Pixel Bender Toolkits zur Verfügung. Kompilieren Sie den Quellcode und speichern Sie die Bytecodedatei unter dem Namen „invertRGB.pbj“ in dasselbe Verzeichnis wie den ActionScript-Quellcode.

Im Folgenden wird der ActionScript-Code für dieses Beispiel angezeigt. Verwenden Sie diese Klasse als Hauptanwendungsklasse für ausschließlich in ActionScript erstellte Projekte in Flash Builder oder als Dokumentklasse für die FLA-Datei in Flash Professional:

```
package
{
    import flash.display.GradientType;
    import flash.display.Graphics;
    import flash.display.Shader;
    import flash.display.Shape;
    import flash.display.Sprite;
    import flash.filters.ShaderFilter;
    import flash.events.Event;
    import flash.geom.Matrix;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class InvertRGB extends Sprite
    {
        private var shader:Shader;
        private var loader:URLLoader;

        public function InvertRGB()
        {
            init();
        }

        private function init():void
        {
            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("invertRGB.pbj"));
        }


        private function onLoadComplete(event:Event):void
        {
```

```
            shader = new Shader(loader.data);

            var target:Shape = new Shape();
            addChild(target);

            var g:Graphics = target.graphics;
            var c:Array = [0x990000, 0x445500, 0x007799];
            var a:Array = [255, 255, 255];
            var r:Array = [0, 127, 255];
            var m:Matrix = new Matrix();
            m.createGradientBox(w, h);
            g.beginGradientFill(GradientType.LINEAR, c, a, r, m);
            g.drawRect(10, 10, w, h);
            g.endFill();

            var invertFilter:ShaderFilter = new ShaderFilter(shader);
            target.filters = [invertFilter];
        }
    }
}
```

Weitere Informationen zur Anwendung von Filtern finden Sie unter „Erstellen und Anwenden von Filtern“ auf Seite 287.

Verwenden eines Shaders im eigenständigen Modus

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wenn Sie einen Shader im Standalone-Modus einsetzen, wird die Shader-Verarbeitung unabhängig davon ausgeführt, wie Sie die Ausgabe verwenden wollen. Sie geben den auszuführenden Shader an, setzen die Eingabe- und Parameterwerte und geben das Objekt an, in das die resultierenden Daten platziert werden. Sie können einen Shader aus zwei Gründen im Standalone-Modus ausführen:

- Verarbeitung von Nicht-Bilddaten: Im Standalone-Modus können Sie anstelle von Bitmap-Bilddaten willkürliche Binär- oder Zahlendaten an den Shader übergeben. Sie können auswählen, ob die Shader-Ergebnisse zusätzlich zu den Bitmap-Bilddaten als Binär- oder Zahlendaten zurückgeben werden.
- Hintergrundverarbeitung: Wenn Sie einen Shader im Standalone-Modus ausführen, wird der Shader standardmäßig asynchron ausgeführt. Dies bedeutet, dass der Shader im Hintergrund ausgeführt wird, während Ihre Anwendung weiter läuft. Ihr Code wird benachrichtigt, wenn die Shader-Verarbeitung abgeschlossen ist. Sie können einen Shader verwenden, der viel Zeit für die Ausführung in Anspruch nimmt und dabei nicht die Benutzeroberfläche der Anwendung oder andere Verabeitungsprozesse blockiert.

Sie verwenden ein ShaderJob-Objekt, um einen Shader im Standalone-Modus auszuführen. Zunächst erstellen Sie das ShaderJob-Objekt und verknüpfen es mit dem Shader-Objekt, das für den auszuführenden Shader steht.

```
var job:ShaderJob = new ShaderJob(myShader);
```

Dann setzen Sie die vom Shader erwarteten Eingabe- oder Parameterwerte. Wenn Sie den Shader im Hintergrund ausführen, registrieren Sie außerdem einen Listener für das `complete`-Ereignis des ShaderJob-Objekt. Ihr Listener wird aufgerufen, wenn der Shader seine Verarbeitung beendet hat.

```
function completeHandler(event:ShaderEvent):void
{
    // do something with the shader result
}

job.addEventListener(ShaderEvent.COMPLETE, completeHandler);
```

Dann erstellen Sie ein Objekt, in das die Ergebnisse des Shaders nach Abschluss der Verarbeitung geschrieben werden. Dieses Objekt weisen Sie der `target`-Eigenschaft des ShaderJob-Objekts zu:

```
var jobResult:BitmapData = new BitmapData(100, 75);
job.target = jobResult;
```

Weisen Sie der `target`-Eigenschaft eine BitmapData-Instanz zu, wenn Sie die Bildverarbeitung mit ShaderJob ausführen. Wenn Sie Binär- oder Zahlendaten verarbeiten, weisen Sie ein ByteArray-Objekt oder eine Vector.<Number>-Instanz für die `target`-Eigenschaft zu. In diesem Fall müssen Sie die Eigenschaften `width` und `height` des ShaderJob-Objekts auf die an das `target`-Objekt auszugebende Datenmenge setzen.

Hinweis: *Sie können die Eigenschaften `shader`, `target`,`width` und `height` des ShaderJob-Objekts in einem Schritt setzen, indem Sie Argumente an den `ShaderJob()`-Konstruktor übergeben (Beispiel: `var job:ShaderJob = new ShaderJob(myShader, myTarget, myWidth, myHeight);`*

Wenn Sie bereit sind, den Shader auszuführen, rufen Sie die `start()`-Methode des ShaderJob-Objekts auf:

```
job.start();
```

Beim Aufruf von `start()` wird ShaderJob standardmäßig asynchron ausgeführt. In diesem Fall fährt die Programmausführung direkt mit der nächsten Codezeile fort und wartet nicht darauf, dass der Shader die Verarbeitung abschließt. Nach Abschluss des Shader-Vorgangs ruft das ShaderJob-Objekt seine `complete`-Ereignis-Listener mit einer entsprechenden Benachrichtigung auf. An diesem Punkt (im Text des `complete`-Ereignis-Listeners) enthält das `target`-Objekt die Ergebnisse des Shader-Vorgangs.

Hinweis: *Anstelle des `target`-Eigenschaftsobjekts können Sie die Shader-Ergebnisse direkt vom Ereignisobjekt abrufen, das an Ihre listener-Methode übergeben wurde. Das Ereignisobjekt ist eine ShaderEvent-Instanz. Das ShaderEvent-Objekt verfügt über drei Eigenschaften, die für den Zugriff auf das Ergebnis verwendet werden können, je nach Datentyp des Objekts, das Sie als `target`-Eigenschaft gesetzt haben: `ShaderEvent.bitmapData`, `ShaderEvent.byteArray` und `ShaderEvent.vector`.*

Alternativ können Sie ein `true`-Argument an die `start()`-Methode senden. In diesem Fall wird der Shader-Vorgang synchron ausgeführt. Während der Ausführung des Shaders wird kein weiterer Code verarbeitet (einschließlich der Interaktion mit der Benutzeroberfläche und anderer Ereignisse). Ist der Shader fertig, enthält das `target`-Objekt die Ergebnisse des Shaders und das Programm fährt mit der nächsten Codezeile fort.

```
job.start(true);
```

Kapitel 16: Verwenden von Movieclips

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die MovieClip-Klasse ist die Hauptklasse für Animationen und Movieclipsymbole, die Sie in der Adobe® Flash®-Entwicklungsumgebung erstellen. Sie verfügt über alle Verhaltensweisen und Funktionen von Anzeigeobjekten, umfasst jedoch zusätzliche Eigenschaften und Methoden zum Steuern der Zeitleiste.

Grundlagen zu Movieclips

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Movieclips sind wichtige Objekte beim Erstellen von animierten Inhalten mit dem Flash-Authoring-Tool und beim Steuern dieser Inhalte mithilfe von ActionScript. Wenn Sie ein Movieclip-Symbol in Flash erstellen, wird das Symbol zur Bibliothek des entsprechenden Flash-Dokuments hinzugefügt. Standardmäßig wird dieses Symbol eine Instanz der MovieClip-Klasse und verfügt deshalb über die Eigenschaften und Methoden der MovieClip-Klasse.

Wenn eine Instanz eines Movieclip-Symbols auf der Bühne positioniert wird, durchläuft der Movieclip automatisch die zugehörige Zeitleiste (bei Movieclips mit mehreren Bildern), es sei denn, die Wiedergabe wird mithilfe von ActionScript geändert. Diese Zeitleiste ist charakteristisch für die MovieClip-Klasse, da Sie damit im Flash-Authoring-Tool Animationen über Bewegungs- oder Form-Tweens erstellen können. Ein Anzeigeobjekt, das eine Instanz der Sprite-Klasse ist, kann im Gegensatz dazu nur animiert werden, wenn die Werte des Objekts programmgesteuert geändert werden.

In früheren Versionen von ActionScript war die MovieClip-Klasse die Basisklasse aller Instanzen auf der Bühne. In ActionScript 3.0 ist ein Movieclip nur eines von vielen Anzeigeobjekten, die auf dem Bildschirm angezeigt werden können. Wenn für die Funktion eines Anzeigeobjekts keine Zeitleiste erforderlich ist, kann die Wiedergabeleistung möglicherweise dadurch gesteigert werden, dass anstelle der MovieClip-Klasse die Shape-Klasse oder die Sprite-Klasse verwendet wird. Weitere Informationen zum Auswählen des geeigneten Anzeigeobjekts für eine Aufgabe finden Sie unter „Auswählen einer DisplayObject-Unterklasse“ auf Seite 184.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe im Zusammenhang mit Movieclips aufgeführt:

AVM1 SWF Eine SWF-Datei, die in der Regel für Flash Player 8 oder frühere Versionen mit ActionScript 1.0 oder ActionScript 2.0 erstellt wurde.

AVM2 SWF Eine SWF-Datei, die mit ActionScript 3.0 für Adobe Flash Player 9 oder höher oder Adobe AIR erstellt wurde.

Externe SWF Eine SWF-Datei, die unabhängig von der Projekt-SWF-Datei erstellt wurde und die in der Projekt-SWF-Datei geladen und wiedergegeben werden soll.

Bild Die kleinste Zeiteinheit in der Zeitleiste. Wie bei einem Filmstreifen eines Spielfilms entspricht jedes Bild einer Momentaufnahme in der Animation. Wenn Bilder in schneller Folge wiedergegeben werden, entsteht der entsprechende Animationseffekt.

Zeitleiste Die abstrahierte Darstellung einer Reihe von Bildern, aus denen sich die Animationssequenz eines Movieclips zusammensetzt. Die Zeitleiste eines MovieClip-Objekts entspricht der Zeitleiste im Flash-Authoring-Tool.

Abspielkopf Eine Markierung, die die Position (das Bild) in der Zeitleiste angibt, die zu einem bestimmten Zeitpunkt wiedergegeben wird.

Verwenden von MovieClip-Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Veröffentlichen einer SWF-Datei werden alle Instanzen eines Movieclip-Symbols auf der Bühne in MovieClip-Objekte umgewandelt. Sie können ein Movieclip-Symbol für ActionScript zur Verfügung stellen, indem Sie diesem Symbol im Feld „Instanzname“ des Eigenschafteninspektors einen Instanznamen zuweisen. Beim Erstellen einer SWF-Datei wird in Flash der Code generiert, mit dem die MovieClip-Instanz auf der Bühne erstellt wird. Außerdem wird eine Variable mit diesem Instanznamen deklariert. Wenn Sie Movieclips benannt haben, die in anderen benannten Movieclips verschachtelt sind, werden diese untergeordneten Movieclips wie Eigenschaften des übergeordneten Movieclips behandelt. Sie können dann über die Punktsyntax auf die untergeordneten Movieclips zugreifen. Wenn beispielsweise der Movieclip mit dem Instanznamen `childClip` in einem anderen Movieclip mit dem Instanznamen `parentClip` verschachtelt ist, kann die Animation auf der Zeitleiste des untergeordneten Movieclips durch Aufrufen des folgenden Codes wiedergegeben werden:

```
parentClip.childClip.play();
```

Hinweis: *: Auf untergeordnete Instanzen, die in der Flash-Authoring-Umgebung auf der Bühne platziert werden, kann nicht durch Code von einem Konstruktor einer übergeordneten Instanz aus zugegriffen werden, da an diesem Punkt der Codeausführung nicht erstellt wurden. Vor dem Zugriff auf die untergeordnete Instanz muss die übergeordnete Instanz die untergeordnete entweder mithilfe von Code erstellen oder den Zugriff auf eine Callback-Funktion verzögern, die überwacht, wann die untergeordnete Instanz ihr Ereignis* `Event.ADDED_TO_STAGE` *auslöst.*

Einige ältere Methoden und Eigenschaften der MovieClip-Klasse von ActionScript 2.0 sind gleich geblieben, andere wiederum wurden geändert. Alle Eigenschaften mit vorangestelltem Unterstrich wurden umbenannt. Die Eigenschaften `_width` und `_height` wurden beispielsweise in `width` und `height` geändert, `_xscale` und `_yscale` in `scaleX` und `scaleY`. Eine vollständige Liste der Eigenschaften und Methoden der MovieClip-Klasse finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Steuern der Wiedergabe von Movieclips

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In Flash werden Animationen oder Statusänderungen mithilfe einer Zeitleiste dargestellt. Bei allen visuellen Elementen mit einer Zeitleiste muss es sich entweder um MovieClip-Objekte oder um Objekte handeln, die von der MovieClip-Klasse abgeleitet sind. Mit ActionScript können Movieclips zwar gestoppt, wiedergegeben oder der Abspielkopf an einen anderen Punkt der Zeitleiste verschoben werden, Zeitleisten können jedoch nicht dynamisch über ActionScript erstellt werden. Zudem können keine Inhalte in spezifische Bilder eingefügt werden. Dies ist nur mit dem Flash-Authoring-Tool möglich.

Bei der Wiedergabe eines MovieClip-Objekts erfolgt der Wiedergabeverlauf entlang der entsprechenden Zeitleiste mit einer Geschwindigkeit, die durch die Bildrate der SWF-Datei vorgegeben ist. Alternativ können Sie diese Einstellung durch Festlegen der `Stage.frameRate`-Eigenschaft in ActionScript überschreiben.

Wiedergeben von Movieclips und Stoppen der Wiedergabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den Methoden `play()` und `stop()` können Movieclips in der entsprechenden Zeitleiste gesteuert werden. Beispiel: Sie erstellen auf der Bühne ein Movieclip-Symbol eines über den Bildschirm fahrenden Fahrrads. Als Instanzname des Movieclip-Symbols geben Sie dabei `bicycle` an. Dann wird der folgende Code mit einem Schlüsselbild auf der Hauptzeitleiste verknüpft:

```
bicycle.stop();
```

Das Fahrrad bewegt sich nicht (die entsprechende Animation wird nicht wiedergegeben). Die Bewegung des Fahrrads kann auch durch eine andere Benutzerinteraktion gestartet werden. Beispielsweise kann mit dem folgenden Code über eine Schaltfläche mit dem Namen `startButton` in einem Schlüsselbild der Hauptzeitleiste festgelegt werden, dass durch Klicken auf diese Schaltfläche die Animation gestartet wird:

```
// This function will be called when the button is clicked. It causes the
// bicycle animation to play.
function playAnimation(event:MouseEvent):void
{
    bicycle.play();
}
// Register the function as a listener with the button.
startButton.addEventListener(MouseEvent.CLICK, playAnimation);
```

Vorspulen und Zurückspulen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Wiedergabe in einem Movieclip kann nicht nur durch die Methoden `play()` und `stop()` gesteuert werden. Sie können den Abspielkopf auch mithilfe der Methoden `nextFrame()` und `prevFrame()` auf der Zeitleiste vor- oder zurückbewegen. Durch den Aufruf einer dieser Methoden werden die Wiedergabe gestoppt und der Abspielkopf jeweils zum nächsten bzw. vorherigen Bild bewegt.

Die Verwendung der `play()`-Methode entspricht dem Aufruf von `nextFrame()` bei jedem Auslösen des `enterFrame`-Ereignisses für ein MovieClip-Objekt. Dementsprechend können Sie den `bicycle`-Movieclip zurückspulen, indem Sie einen Ereignis-Listener für das `enterFrame`-Ereignis erstellen und `bicycle` wie folgt auf das vorherige Bild in der Listener-Funktion zurücksetzen:

```
// This function is called when the enterFrame event is triggered, meaning
// it's called once per frame.
function everyFrame(event:Event):void
{
    if (bicycle.currentFrame == 1)
    {
        bicycle.gotoAndStop(bicycle.totalFrames);
    }
    else
    {
        bicycle.prevFrame();
    }
}
bicycle.addEventListener(Event.ENTER_FRAME, everyFrame);
```

Wenn ein Movieclip mehrere Bilder enthält, wird er bei der normalen Wiedergabe endlos durchlaufen, d. h. er wird nach dem letzten Bild wieder auf Bild 1 zurückgesetzt. Bei Verwendung von `prevFrame()` oder `nextFrame()` erfolgt dieses Verhalten nicht automatisch (wenn sich der Abspielkopf beim Aufruf von `prevFrame()` auf Bild 1 befindet, wird er nicht zum letzten Bild bewegt). Mit der `if`-Bedingung in diesem Beispiel wird überprüft, ob der Abspielkopf rückwärts zum ersten Bild bewegt wurde. In diesem Fall wird der Abspielkopf vorwärts zum letzten Bild bewegt, sodass eine Endlosschleife für den Movieclip entsteht, in der er rückwärts wiedergegeben wird.

Springen zu anderen Bildern und Verwenden von Bildbeschriftungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Movieclip kann ganz einfach auf ein neues Bild gesetzt werden. Durch Aufrufen von `gotoAndPlay()` oder `gotoAndStop()` springt der Movieclip zu der Bildnummer, die als Parameter angegeben ist. Alternativ können Sie einen String angeben, der dem Namen einer Bildbeschriftung entspricht. Jedem Bild in der Zeitleiste kann eine Beschriftung zugewiesen werden. Wählen Sie dazu ein Bild in der Zeitleiste aus und geben Sie dann im Eigenschafteninspektor im Feld „Bildbeschriftung“ einen Namen ein.

Die Vorteile der Verwendung von Bildbeschriftungen anstelle von Bildnummern machen sich beim Erstellen komplexer Movieclips besonders bemerkbar. Wenn eine Animation viele Bilder, Ebenen und Tweens enthält, sollten Sie wichtige Bilder mit Beschriftungen versehen, die Änderungen in der Animation des Movieclips angeben (z. B. „off“, „walking“ oder „running“). Dies verbessert die Lesbarkeit des Codes und ermöglicht zudem eine flexible Verwendung, da es sich bei ActionScript-Aufrufen für ein beschriftetes Bild um Zeiger auf einen einzelnen Verweis (die Beschriftung) und nicht auf eine bestimmte Bildnummer handelt. Wenn Sie zu einem späteren Zeitpunkt ein bestimmtes Animationssegment in ein anderes Bild verschieben möchten, müssen Sie den ActionScript-Code nur ändern, wenn Sie für die Bilder an der neuen Position andere Beschriftungen verwenden.

Mithilfe der FrameLabel-Klasse von ActionScript 3.0 können Bildbeschriftungen im Code verwendet werden. Jede Instanz dieser Klasse repräsentiert jeweils eine Bildbeschriftung und verfügt über eine `name`-Eigenschaft für den Namen der Bildbeschriftung im Eigenschafteninspektor und über eine `frame`-Eigenschaft für die Bildnummer des Bilds, für das die Beschriftung auf der Zeitleiste positioniert wird.

Damit die mit einer MovieClip-Instanz verknüpften FrameLabel-Instanzen abgerufen werden können, enthält die MovieClip-Klasse zwei Eigenschaften, die direkt FrameLabel-Objekte zurückgeben. Mit der `currentLabels`-Eigenschaft wird ein Array zurückgegeben, das alle FrameLabel-Objekte in der gesamten Zeitleiste eines Movieclips enthält. Mit der `currentLabel`-Eigenschaft wird ein String zurückgegeben, der den Namen der zuletzt auf der Zeitleiste angetroffenen Bildbeschriftung enthält.

Wenn Sie einen Movieclip mit dem Namen `robot` erstellen und die verschiedenen Animationsbewegungen mit Beschriftungen versehen haben, können Sie beispielsweise eine Bedingung festlegen, mit der die `currentLabel`-Eigenschaft überprüft wird, um die aktuelle Bewegung von `robot` abzurufen, wie im folgenden Codebeispiel dargestellt:

```
if (robot.currentLabel == "walking")
{
    // do something
}
```

In Flash Player 11.3 und AIR 3.3 wurde das `frameLabel`-Ereignis zur FrameLabel-Klasse hinzugefügt. Sie können der FrameLabel-Instanz, die eine Bildbeschriftung darstellt, eine Ereignisprozedur zuweisen. Das Ereignis wird abgesetzt, wenn der Abspielkopf in das Bild eintritt.

Im folgenden Beispiel wird eine FrameLabel-Instanz für die zweite Bildbeschriftung im Array mit Bildbeschriftungen für den MovieClip erstellt. Dann wird eine Ereignisprozedur für das `frameLabel`-Ereignis registriert:

```
var myFrameLabel:FrameLabel = robot.currentLabels[1];
myFrameLabel.addEventListener(Event.FRAME_LABEL, onFrameLabel);

function onFrameLabel(e:Event):void {
    //do something
}
```

Arbeiten mit Szenen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Flash-Authoring-Umgebung können Sie mithilfe von Szenen mehrere Zeitleisten festlegen, entlang derer eine SWF-Datei wiedergegeben wird. Über den zweiten Parameter der `gotoAndPlay()`-Methode oder der `gotoAndStop()`-Methode können Sie eine Szene angeben, zu der der Abspielkopf bewegt wird. Alle FLA-Dateien beginnen lediglich mit der Anfangsszene, Sie können jedoch neue Szenen erstellen.

Die Verwendung von Szenen ist nicht immer die am besten geeignete Methode, da dies eine Reihe von Nachteilen mit sich bringt. Ein Flash-Dokument mit mehreren Szenen ist unter Umständen schwer zu verwalten, vor allem in Umgebungen mit mehreren Programmierern. Die Verwendung mehrerer Szenen kann sich zudem negativ auf die Bandbreite auswirken, da beim Veröffentlichen alle Szenen in einer einzigen Zeitleiste zusammengeführt werden. Dies führt zu einem progressiven Download aller Szenen, selbst wenn diese nie wiedergegeben werden. Aus diesen Gründen wird von der Verwendung mehrerer Szenen häufig abgeraten, es sei denn zum Strukturieren mehrerer Animationen mit Zeitleisten.

Mit der `scenes`-Eigenschaft der MovieClip-Klasse wird ein Array mit Scene-Objekten zurückgegeben, in dem alle Szenen in der SWF-Datei angegeben sind. Mit der `currentScene`-Eigenschaft wird ein Scene-Objekt für die derzeit wiedergegebene Szene zurückgegeben.

Die Scene-Klasse verfügt über mehrere Eigenschaften, die Informationen zu einer Szene bereitstellen. Die `labels`-Eigenschaft gibt ein Array mit FrameLabel-Objekten mit den Bildbeschriftungen in dieser Szene zurück. Mit der `name`-Eigenschaft wird der Name der Szene als String zurückgegeben. Mit der `numFrames`-Eigenschaft wird eine Ganzzahl zurückgegeben, die die Gesamtanzahl der Bilder in einer Szene angibt.

Erstellen von MovieClip-Objekten mit ActionScript

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Inhalte können in Flash auf dem Bildschirm hinzugefügt werden, indem Elemente von der Bibliothek auf die Bühne gezogen werden. Dies ist jedoch nicht das einzig mögliche Verfahren. Bei komplexen Projekten bevorzugen erfahrene Entwickler in der Regel die programmgesteuerte Erstellung von Movieclips. Dieser Ansatz hat mehrere Vorteile: leichtere Wiederverwendbarkeit des Codes, gesteigerte Kompiliergeschwindigkeit sowie bessere Änderungsmöglichkeiten, die nur in ActionScript verfügbar sind.

Die Anzeigelisten-API von ActionScript 3.0 vereinfacht den Vorgang der dynamischen Erstellung von MovieClip-Objekten. Die Möglichkeit der direkten Instanziierung von MovieClip-Instanzen, unabhängig vom Hinzufügen dieser Instanzen zur Anzeigeliste, bietet Flexibilität und Einfachheit ohne Abstriche bei der Steuerung.

Wenn Sie in ActionScript 3.0 eine MovieClip-Instanz (oder ein Instanz eines anderen Anzeigeobjekts) programmgesteuert erstellen, wird diese erst auf dem Bildschirm sichtbar, nachdem sie durch Aufrufen der `addChild()`- oder der `addChildAt()`-Methode für einen Anzeigeobjektcontainer zur Anzeigeliste hinzugefügt wurde. Dadurch können Sie einen Movieclip erstellen, die zugehörigen Eigenschaften und sogar die zugehörigen Methoden aufrufen, bevor der Movieclip auf dem Bildschirm wiedergegeben wird. Weitere Informationen zum Arbeiten mit der Anzeigeliste finden Sie unter „Arbeiten mit Anzeigeobjektcontainern“ auf Seite 170.

Exportieren von Bibliothekssymbolen für ActionScript

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Standardmäßig können Instanzen von Movieclip-Symbolen in der Bibliothek eines Flash-Dokuments nicht dynamisch (d. h. nur mit ActionScript) erstellt werden. Das liegt daran, dass jedes Symbol, das zur Verwendung in ActionScript exportiert wird, zur Vergrößerung der SWF-Datei beiträgt und dass einige Symbole möglicherweise nicht zur Verwendung auf der Bühne vorgesehen sind. Aus diesem Grund muss angegeben werden, dass ein Symbol in ActionScript exportiert werden soll, damit dieses Symbol in ActionScript zur Verfügung steht.

So exportieren Sie ein Symbol für ActionScript:

1. Wählen Sie das Symbol im Bedienfeld „Bibliothek“ aus und öffnen Sie das entsprechende Dialogfeld „Symboleigenschaften“.
2. Aktivieren Sie gegebenenfalls die erweiterten Einstellungen.
3. Aktivieren Sie im Bereich „Verknüpfung“ das Kontrollkästchen „Export für ActionScript“.

   Dadurch werden die Felder „Klasse“ und „Basisklasse“ aktiviert.

   In der Standardeinstellung wird im Feld „Klasse“ der Symbolname übernommen, die Leerzeichen werden dabei entfernt (der Symbolname „Tree House“ wird beispielsweise in „TreeHouse“ geändert). Wenn Sie angeben möchten, dass das Verhalten des Symbols mit einer benutzerdefinierten Klasse festgelegt werden soll, geben Sie in diesem Feld den vollständigen Namen der entsprechenden Klasse mit Angabe des zugehörigen Pakets an. Wenn Sie Instanzen des Symbols in ActionScript erstellen, jedoch kein weiteres Verhalten hinzufügen möchten, können Sie den Klassennamen beibehalten.

   Im Feld „Basisklasse“ ist automatisch der Wert `flash.display.MovieClip` eingetragen. Wenn das Symbol um die Funktionen einer anderen benutzerdefinierten Klasse erweitert werden soll, können Sie stattdessen den Namen der entsprechenden Klasse eingeben. Diese Klasse muss die Sprite-Klasse (oder MovieClip-Klasse) erweitern.
4. Klicken Sie auf die Schaltfläche „OK“, um die Änderungen zu speichern.

   Wenn zu diesem Zeitpunkt in Flash keine verknüpfte SWC-Datei oder eine externe ActionScript-Datei mit einer Definition der angegebenen Klasse gefunden wird (wenn Sie beispielsweise kein weiteres Verhalten für das Symbol angegeben haben), wird die folgende Warnmeldung angezeigt:

   Im Klassenpfad konnte keine Definition für diese Klasse gefunden werden, daher wird in der SWF-Datei beim Export automatisch eine generiert.

   Sie können diese Warnmeldung ignorieren, wenn für das entsprechende Bibliothekssymbol zusätzlich zu den Funktionen der MovieClip-Klasse keine weiteren eindeutigen Funktionen erforderlich sind.

Wenn Sie keine Klasse für das Symbol angeben, wird in Flash eine Klasse erstellt, die der folgenden Klasse entspricht:

```
package
{
    import flash.display.MovieClip;

    public class ExampleMovieClip extends MovieClip
    {
        public function ExampleMovieClip()
        {
        }
    }
}
```

Wenn das Symbol über erweiterte ActionScript-Funktionen verfügen soll, fügen Sie die entsprechenden Eigenschaften und Methoden zur folgenden Codestruktur hinzu. Beispiel: Bei einem Movieclip-Symbol mit einem Kreis mit einer Breite und Höhe von 50 Pixel ist die Option „Export für ActionScript“ aktiviert und als Circle-Klasse angegeben. Durch Einfügen des folgenden Codes in die Datei „Circle.as“ werden die MovieClip-Klasse erweitert und das Symbol mit den zusätzlichen Methoden `getArea()` und `getCircumference()` versehen:

```
package
{
    import flash.display.MovieClip;

    public class Circle extends MovieClip
    {
        public function Circle()
        {
        }

        public function getArea():Number
        {
        // The formula is Pi times the radius squared.
        return Math.PI * Math.pow((width / 2), 2);
        }

        public function getCircumference():Number
        {
        // The formula is Pi times the diameter.
        return Math.PI * width;
        }
    }
}
```

Mit dem folgenden Code, der in einem Schlüsselbild in Bild 1 des Flash-Dokuments eingefügt wird, wird eine Instanz des Symbols erstellt und auf dem Bildschirm angezeigt:

```
var c:Circle = new Circle();
addChild(c);
trace(c.width);
trace(c.height);
trace(c.getArea());
trace(c.getCircumference());
```

Mit diesem Code wird die ActionScript-Instanziierung als Alternative zum Ziehen einzelner Elemente auf die Bühne veranschaulicht. Es wird ein Kreis erstellt, der über alle Eigenschaften eines Movieclips und zusätzlich über die benutzerdefinierten Methoden verfügt, die in der Circle-Klasse definiert sind. Hierbei handelt es sich um ein sehr einfaches Beispiel. In der Klasse eines Bibliothekssymbols können beliebig viele Eigenschaften und Methoden festgelegt werden.

Die ActionScript-Instanziierung ist sehr leistungsstark, da Sie eine große Anzahl Instanzen dynamisch erstellen können, deren manuelle Anordnung ein sehr langwieriges Unterfangen darstellt. Diese Instanziierung ist zudem flexibel, da Sie die Eigenschaften jeder Instanz beim Erstellen anpassen können. Diese Vorteile machen sich bemerkbar, wenn mehrere Circle-Instanzen mithilfe einer Schleife dynamisch erstellt werden. Wenn sich das zuvor beschriebene Circle-Symbol und die Circle-Klasse in der Bibliothek des Flash-Dokuments befinden, fügen Sie den folgenden Code in einem Schlüsselbild in Bild 1 ein:

```
import flash.geom.ColorTransform;

var totalCircles:uint = 10;
var i:uint;
for (i = 0; i < totalCircles; i++)
{
    // Create a new Circle instance.
    var c:Circle = new Circle();
    // Place the new Circle at an x coordinate that will space the circles
    // evenly across the Stage.
    c.x = (stage.stageWidth / totalCircles) * i;
    // Place the Circle instance at the vertical center of the Stage.
    c.y = stage.stageHeight / 2;
    // Change the Circle instance to a random color
    c.transform.colorTransform = getRandomColor();
    // Add the Circle instance to the current timeline.
    addChild(c);
}

function getRandomColor():ColorTransform
{
    // Generate random values for the red, green, and blue color channels.
    var red:Number = (Math.random() * 512) - 255;
    var green:Number = (Math.random() * 512) - 255;
    var blue:Number = (Math.random() * 512) - 255;

    // Create and return a ColorTransform object with the random colors.
    return new ColorTransform(1, 1, 1, 1, red, green, blue, 0);
}
```

Hierdurch wird veranschaulicht, wie Sie im Code schnell mehrere Instanzen eines Symbols erstellen und anpassen können. Jede Instanz wird anhand des aktuellen Werts des Schleifenzählers positioniert und erhält über die `transform`-Eigenschaft (die die Circle-Klasse durch Erweiterung der MovieClip-Klasse erbt) eine zufällig gewählte Farbe.

Laden einer externen SWF-Datei

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 3.0 werden SWF-Dateien mit der Loader-Klasse geladen. Zum Laden einer externen SWF-Datei müssen im ActionScript-Code die folgenden vier Vorgänge durchgeführt werden:

1. Erstellen eines neuen URLRequest-Objekts mit der URL der Datei
2. Erstellen eines neuen Loader-Objekts
3. Aufrufen der `load()`-Methode des Loader-Objekts und Übergeben der URLRequest-Instanz als Parameter

4 Aufrufen der `addChild()`-Methode für einen Anzeigeobjektcontainer (z. B. die Hauptzeitleiste eines Flash-Dokuments) zum Hinzufügen der Loader-Instanz zur Anzeigeliste

Dies ergibt schließlich den folgenden Code:

```
var request:URLRequest = new URLRequest("http://www.[yourdomain].com/externalSwf.swf");
var loader:Loader = new Loader()
loader.load(request);
addChild(loader);
```

Mit dem gleichen Code kann anstelle einer SWF-Datei eine externe Bilddatei (z. B. eine JPEG-, GIF- oder PNG-Datei) geladen werden, indem Sie die URL der Bilddatei angeben. Im Gegensatz zu einer Bilddatei kann eine SWF-Datei ActionScript-Code enthalten. Obwohl eine SWF-Datei wie eine Bilddatei geladen wird, müssen sich beim Laden einer externen SWF-Datei die ladende SWF-Datei und die geladene SWF-Datei in der gleichen Sicherheits-Sandbox befinden, wenn die SWF-Datei von Flash Player oder AIR wiedergegeben wird und Sie mithilfe von ActionScript eine Verbindung mit der externen SWF-Datei herstellen möchten. Wenn die externe SWF-Datei Klassen enthält, die über den gleichen Namespace verfügen wie Klassen in der ladenden SWF-Datei, müssen Sie unter Umständen eine neue Anwendungsdomäne für die geladene SWF-Datei erstellen, damit keine Namespace-Konflikte auftreten. Weitere Informationen zur Sicherheit und zu Anwendungsdomänen finden Sie unter „Verwenden von Anwendungsdomänen“ auf Seite 157 und „Laden von Inhalten“ auf Seite 1135.

Wenn die externe SWF-Datei erfolgreich geladen wurde, kann sie über die `Loader.content`-Eigenschaft aufgerufen werden. Wenn die externe SWF-Datei für ActionScript 3.0 veröffentlicht wird, handelt es sich je nachdem, welche Klasse erweitert wird, um einen Movieclip oder um ein Sprite.

Es bestehen einige Unterschiede zwischen dem Laden einer SWF-Datei in Adobe AIR für iOS und anderen Plattformen. Weitere Informationen finden Sie unter „Laden von SWF-Dateien in AIR für iOS“ auf Seite 215.

Hinweise zum Laden älterer SWF-Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn eine externe SWF-Datei mit einer älteren ActionScript-Version veröffentlicht wurde, müssen einige wichtige Beschränkungen beachtet werden. Im Gegensatz zu einer für ActionScript 3.0 veröffentlichten SWF-Datei, die in AVM2 (ActionScript Virtual Machine 2) ausgeführt wird, wird eine für ActionScript 1.0 oder 2.0 veröffentlichte Datei in AVM1 (ActionScript Virtual Machine 1) ausgeführt.

Es gibt wichtige Unterschiede beim Laden einer ActionScript 1.0- oder 2.0-SWF-Datei in eine ActionScript 3.0-SWF-Datei (im Vergleich zum Laden einer ActionScript 3.0-SWF-Datei). Flash Player bietet vollständige Abwärtskompatibilität mit bereits veröffentlichten Inhalten. Inhalt, der in früheren Flash Player-Versionen ausgeführt wird, läuft auch in Flash Player-Versionen, die ActionScript 3.0 unterstützen. Es gelten jedoch die folgenden Einschränkungen:

- ActionScript 3.0-Code kann eine SWF-Datei laden, die in ActionScript 1.0 oder 2.0 geschrieben wurde. Wenn eine ActionScript 1.0- oder 2.0-SWF-Datei erfolgreich geladen wird, ist das geladene Objekt (die `Loader.content`-Eigenschaft) ein AVM1Movie-Objekt. Bei einer AVM1Movie-Instanz und einer MovieClip-Instanz handelt es sich um unterschiedliche Instanzen. Eine AVM1Movie-Instanz ist ein Anzeigeobjekt. Im Gegensatz zu einem Movieclip enthält es keine zeitleistenspezifischen Methoden oder Eigenschaften. Die übergeordnete AVM2-SWF-Datei kann nicht auf die Eigenschaften, Methoden oder Objekte des geladenen AVM1Movie-Objekts zugreifen.
- In ActionScript 1.0 oder 2.0 geschriebene SWF-Dateien können keine in ActionScript 3.0 geschriebenen SWF-Dateie nladen. Dies bedeutet, dass in Flash 8, Flex Builder 1.5 oder älteren Versionen erstellte SWF-Dateien keine mit ActionScript 3.0 erstellten SWF-Dateien laden können.

Die einzige Ausnahme dieser Regel besteht darin, dass ActionScript 2.0-SWF-Dateien sich selbst durch eine ActionScript 3.0-SWF-Datei ersetzen können, falls zuvor noch keine Daten geladen wurden. Dies ist möglich, wenn die ActionScript 2.0-SWF-Datei `loadMovieNum()` aufruft und dabei im Parameter `level` den Wert 0 übergibt.

- Im Allgemeinen müssen in ActionScript 1.0 oder 2.0 geschriebene SWF-Dateien migriert werden, wenn sie zusammen mit in ActionScript 3.0 geschriebenen SWF-Dateien verwendet werden sollen. Angenommen beispielsweise, Sie haben einen Media Player mit ActionScript 2.0 erstellt. Der Media Player lädt verschiedene Inhalte, die ebenfalls mit ActionScript 2.0 erstellt wurden. Sie können keine neuen Inhalte in ActionScript 3.0 erstellen und diese in den Media Player laden. Dazu müssen Sie den Media Player zu ActionScript 3.0 migrieren.

  Wenn Sie dagegen einen Media Player in ActionScript 3.0 erstellen, kann dieser einfache Ladevorgänge für ActionScript 2.0-Inhalte durchführen.

In den folgenden Tabellen sind die Einschränkungen älterer Versionen von Flash Player bezüglich des Ladens neuerer Inhalte und des Ausführens von Code aufgeführt. Außerdem enthalten sie die Einschränkungen bei der Skriptreferenzierung zwischen SWF-Dateien mit unterschiedlichen ActionScript-Versionen.

Unterstützte Funktionen	**Flash Player 7**	**Flash Player 8**	**Flash Player 9 und 10**
Laden von SWF-Dateien für	7 und früher	8 und früher	9 (oder 10) und früher
Enthält folgende AVM	AVM1	AVM1	AVM1 und AVM2
Ausführen von SWF-Dateien mit ActionScript	1.0 und 2.0	1.0 und 2.0	1.0, 2.0 und 3.0

In der folgenden Tabelle bezieht sich der Ausdruck „Unterstützte Funktionen“ auf Inhalte, die in Flash Player 9 oder höher ausgeführt werden. In Flash Player bis Version 8 ausgeführte Inhalte können nur ActionScript 1.0 und 2.0 laden, anzeigen, ausführen sowie entsprechende Skripts in anderen SWF-Dateien referenzieren.

Unterstützte Funktionen	**In ActionScript 1.0 und 2.0 erstellte Inhalte**	**In ActionScript 3.0 erstellte Inhalte**
Laden von Inhalten und Ausführen von Code in Inhalten in	ActionScript 1.0 und 2.0 (ausschließlich)	ActionScript 1.0, 2.0 und ActionScript 3.0
Referenzieren von anderen Skripts in	ActionScript 1.0 und 2.0 (ausschließlich; ActionScript 3.0 über lokale Verbindungen)	ActionScript 1.0 und 2.0 über lokale Verbindungen ActionScript 3.0

Movieclip-Beispiel: RuntimeAssetsExplorer

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Funktion „Export für ActionScript“ ist besonders nützlich bei Bibliotheken, die möglicherweise in mehreren Projekten verwendet werden. Wenn Flash Player oder AIR eine SWF-Datei ausführt, stehen nach ActionScript exportierte Symbole jeder SWF-Datei zur Verfügung, die sich in der gleichen Sicherheits-Sandbox befindet wie die ladende SWF-Datei. Auf diese Weise kann in einem Flash-Dokument eine SWF-Datei erstellt werden, die nur zum Speichern grafischer Elemente dient. Dieses Verfahren eignet sich besonders für größere Projekte, bei denen Designer, die visuelle Elemente bearbeiten, parallel mit Entwicklern arbeiten, die eine „umhüllende“ SWF-Datei erstellen, über die dann die SWF-Datei mit den grafischen Elementen zur Laufzeit geladen wird. Mit diesem Verfahren können Sie mehrere Dateiversionen verwalten, in denen grafische Elemente nicht von der Programmierentwicklung abhängen.

Mit der Anwendung „RuntimeAssetsExplorer“ werden alle SWF-Dateien geladen, die Unterklassen der RuntimeAsset-Klasse sind. Mit dieser Anwendung können Sie zudem die verfügbaren Elemente dieser SWF-Dateien durchsuchen. Mit diesem Beispiel wird Folgendes veranschaulicht:

- Laden einer externen SWF-Datei mit `Loader.load()`
- Dynamisches Erstellen eines in ActionScript exportierten Bibliothekssymbols
- Steuern der MovieClip-Wiedergabe über ActionScript

Beachten Sie zunächst, dass sich alle in Flash Player auszuführenden SWF-Dateien in der gleichen Sicherheits-Sandbox befinden müssen. Weitere Informationen finden Sie unter „Sicherheits-Sandboxen“ auf Seite 1117.

Die Anwendungsdateien für dieses Beispiel stehen über die Flash Professional-Beispiele zum Download bereit. Die Dateien der Anwendung „RuntimeAssetsExplorer“ befinden sich im Ordner „Samples/RuntimeAssetsExplorer“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
RuntimeAssetsExample.mxml oder RuntimeAssetsExample.fla	Die Benutzeroberfläche der Anwendung im Flex-Format (MXML) oder Flash-Format (FLA).
RuntimeAssetsExample.as	Document-Klasse für die Flash-Anwendung (FLA).
GeometricAssets.as	Eine Beispielklasse für die Implementierung der RuntimeAsset-Schnittstelle.
GeometricAssets.fla	Eine FLA-Datei, die mit der GeometricAssets-Klasse (die Dokumentklasse der FLA-Datei) verknüpft ist und Symbole enthält, die für ActionScript exportiert werden.
com/example/programmingas3/runtimeassetexplorer/RuntimeLibrary.as	Eine Schnittstelle, in der die erforderlichen Methoden aller SWF-Dateien mit Laufzeitelementen definiert sind, die im Explorer-Container geladen werden.
com/example/programmingas3/runtimeassetexplorer/AnimatingBox.as	Die Klasse des Bibliothekssymbols in Form eines sich drehenden Feldes.
com/example/programmingas3/runtimeassetexplorer/AnimatingStar.as	Die Klasse des Bibliothekssymbols in Form eines sich drehenden Sterns.

Erstellen einer Schnittstelle für die Laufzeitbibliothek

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Damit über den Explorer ordnungsgemäß auf eine SWF-Bibliothek zugegriffen werden kann, muss die Struktur der Laufzeit-Elementbibliotheken formalisiert werden. Dies erfolgt durch Erstellen einer Schnittstelle. Diese ähnelt einer Klasse, da sie einen Entwurf der Methoden enthält und die zu erwartende Struktur festlegt. Im Gegensatz zu einer Klasse wird jedoch nicht der Methodenrumpf angegeben. Über die Schnittstelle kann eine Verbindung zwischen der Laufzeitbibliothek und dem Explorer hergestellt werden. Jede SWF-Datei mit Laufzeitelementen, die im Browser geladen wird, implementiert diese Schnittstelle. Weitere Informationen zu Schnittstellen und ihren Verwendungsmöglichkeiten finden Sie im Abschnitt zu Schnittstellen im *ActionScript 3.0 – Arbeitshandbuch*.

Die RuntimeLibrary-Schnittstelle ist sehr einfach. Es wird lediglich eine Funktion benötigt, über die im Explorer ein Array mit Klassenpfaden verfügbar ist, damit die Symbole exportiert werden und in der Laufzeitbibliothek zur Verfügung stehen. Zu diesem Zweck enthält die Schnittstelle eine einzige Methode: `getAssets()`.

```
package com.example.programmingas3.runtimeassetexplorer
{
    public interface RuntimeLibrary
    {
        function getAssets():Array;
    }
}
```

Erstellen der SWF-Datei für die Elementbibliothek

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Definition der RuntimeLibrary-Schnittstelle können mehrere SWF-Dateien für die Elementbibliothek erstellt werden, die in eine andere SWF-Datei geladen werden können. Die Erstellung einer SWF-Datei für die Elementbibliothek umfasst vier Aufgaben:

- Erstellen einer Klasse für die SWF-Datei der Elementbibliothek
- Erstellen von Klassen für einzelne Elemente in der Bibliothek
- Erstellen der eigentlichen grafischen Elemente
- Verknüpfen der grafischen Elemente mit Klassen und Veröffentlichen der SWF-Datei für die Bibliothek

Erstellen einer Klasse zum Implementieren der RuntimeLibrary-Schnittstelle

Im Folgenden wird die GeometricAssets-Klasse erstellt, mit der die RuntimeLibrary-Schnittstelle implementiert wird. Dabei handelt es sich um die Dokumentklasse der FLA-Datei. Der Code für diese Klasse ähnelt dem Code für die RuntimeLibrary-Schnittstelle mit dem Unterschied, dass in der Klassendefinition der `getAssets()`-Methode der Methodenrumpf angegeben ist.

```
package
{
    import flash.display.Sprite;
    import com.example.programmingas3.runtimeassetexplorer.RuntimeLibrary;

    public class GeometricAssets extends Sprite implements RuntimeLibrary
    {
        public function GeometricAssets() {

        }
        public function getAssets():Array {
            return [ "com.example.programmingas3.runtimeassetexplorer.AnimatingBox",
                     "com.example.programmingas3.runtimeassetexplorer.AnimatingStar" ];
        }
    }
}
```

Im Fall der Erstellung einer zweiten Laufzeitbibliothek kann eine weitere FLA-Datei auf der Grundlage einer anderen Klasse (z. B. „AnimationAssets“) erstellt werden, die wiederum eine eigene Implementierung von `getAssets()` enthält.

Erstellen von Klassen für jedes MovieClip-Element

In diesem Beispiel wird die MovieClip-Klasse lediglich erweitert, ohne dass weitere Funktionen zu den benutzerdefinierten Elementen hinzugefügt werden. Der folgende Code für AnimatingStar ist analog zum Code für AnimatingBox:

```
package com.example.programmingas3.runtimeassetexplorer
{
    import flash.display.MovieClip;

    public class AnimatingStar extends MovieClip
    {
        public function AnimatingStar() {
        }
    }
}
```

Veröffentlichen der Bibliothek

Die MovieClip-Elemente werden nun mit der neuen Klasse verknüpft. Dazu wird eine neue FLA-Datei erstellt und im Eigenschafteninspektor wird im Feld „Dokumentklasse“ die GeometricAssets-Klasse eingegeben. In diesem Beispiel werden zwei einfache Formen erstellt, die mit einem Zeitleisten-Tween über 360 Bilder einmal im Uhrzeigersinn gedreht werden. Für die beiden Symbole `animatingBox` und `animatingStar` wird die Option „Export für ActionScript“ aktiviert und im Feld „Klasse“ jeweils der entsprechende in der `getAssets()`-Implementierung angegebene Klassenpfad eingegeben. Die Standardbasisklasse von `flash.display.MovieClip` wird beibehalten, da Unterklassen der MovieClip-Standardmethoden erstellt werden sollen.

Legen Sie die Exporteinstellungen für das Symbol fest und veröffentlichen Sie dann die FLA-Datei. Damit ist die erste Laufzeitbibliothek erstellt. Diese SWF-Datei kann beispielsweise in einer anderen AVM2-SWF-Datei geladen werden. Das AnimatingBox-Symbol und das AnimatingStar-Symbol sind dann in der neuen SWF-Datei verfügbar.

Laden der Bibliothek in eine andere SWF-Datei

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In einem letzten Schritt muss die Benutzeroberfläche für den Elemente-Explorer erstellt werden. In diesem Beispiel ist der Pfad für die Laufzeitbibliothek in der Variablen `ASSETS_PATH` hartkodiert (fest vorgegeben). Alternativ können Sie mithilfe der FileReference-Klasse beispielsweise eine Benutzeroberfläche erstellen, über die nach einer bestimmten SWF-Datei auf der Festplatte gesucht wird.

Nach dem erfolgreichen Laden der Laufzeitbibliothek wird die `runtimeAssetsLoadComplete()`-Methode in Flash Player aufgerufen:

```
private function runtimeAssetsLoadComplete(event:Event):void
{
    var rl:* = event.target.content;
    var assetList:Array = rl.getAssets();
    populateDropdown(assetList);
    stage.frameRate = 60;
}
```

In dieser Methode wird die geladene SWF-Datei durch die Variable „rl“ dargestellt. Im Code wird die `getAssets()`-Methode der geladenen SWF-Datei aufgerufen; damit wird die Liste der verfügbaren Elemente abgerufen. Anhand dieser Elemente wird durch Aufrufen der `populateDropDown()`-Methode eine ComboBox-Komponente mit der Liste verfügbarer Elemente gefüllt. Mit dieser Methode wird wiederum der vollständige Klassenpfad jedes Elements gespeichert. Durch Klicken auf die Schaltfläche „Add“ der Benutzeroberfläche wird die `addAsset()`-Methode ausgelöst:

```
private function addAsset():void
{
    var className:String = assetNameCbo.selectedItem.data;
    var AssetClass:Class = getDefinitionByName(className) as Class;
    var mc:MovieClip = new AssetClass();
    ...
}
```

Dadurch werden der Klassenpfad des derzeit in der ComboBox-Komponente (`assetNameCbo.selectedItem.data`) ausgewählten Elements sowie mithilfe der `getDefinitionByName()`-Funktion (aus dem flash.utils-Paket) ein Verweis auf die Klasse des Elements abgerufen, sodass eine neue Instanz des Elements erstellt wird.

Kapitel 17: Arbeiten mit Bewegungs-Tweens

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Unter „Animieren von Objekten“ auf Seite 205 wird beschrieben, wie Sie Animationsskripts in ActionScript implementieren.

In diesem Kapitel wird eine andere Technik zur Erstellung von Animationen beschrieben: Bewegungs-Tweens. Diese Technik ermöglicht es Ihnen, Bewegungen zu erstellen, indem Sie die Bewegung mithilfe von Adobe® Flash® Professional interaktiv in einem Dokument einrichten. Anschließend können Sie diese Bewegung zur Laufzeit in dynamischen ActionScript-Animationen verwenden.

Flash Professional generiert automatisch den ActionScript-Code, der das Bewegungs-Tween implementiert, sodass Sie den Code kopieren und wieder verwenden können.

Zur Erstellung von Bewegungs-Tweens benötigen Sie eine Lizenz für Flash Professional.

Verwandte Hilfethemen

fl.motion-Paket

Grundlagen von Bewegungs-Tweens

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Bewegungs-Tweens bieten eine einfache Möglichkeit zur Erstellung von Animationen.

Ein Bewegungs-Tween ändert die Eigenschaften von Anzeigeobjekten, wie beispielsweise Position oder Drehung, von Bild zu Bild. Außerdem kann ein Bewegungs-Tween die Darstellung eines Anzeigeobjekts während der Bewegung ändern, indem verschiedene Filter und andere Eigenschaften angewendet werden. Sie erstellen das Bewegungs-Tween interaktiv in Flash Professional. Dabei wird der ActionScript-Code für das Bewegungs-Tween automatisch generiert. Wählen Sie in Flash den Befehl „Bewegung als ActionScript 3.0 kopieren“, um den ActionScript-Code zu kopieren, mit dem das Bewegungs-Tween erstellt wurde. Diesen ActionScript-Code können Sie dann wieder verwenden, um in Ihren eigenen dynamischen Animationen zur Laufzeit Bewegungen zu erstellen.

Weitere Informationen zum Erstellen eines Bewegungs-Tweens finden Sie im Abschnitt zu Bewegungs-Tweens in der Dokumentation *Verwenden von Flash Professional*.

Wichtige Konzepte und Begriffe

Folgendes ist ein wichtiger Begriff, der für diese Funktion relevant ist:

Bewegungs-Tween Ein Konstrukt, das Zwischenbilder eines Anzeigeobjekts in verschiedenen Zuständen zu unterschiedlichen Zeitpunkten generiert. Dabei entsteht der Eindruck, dass der erste Zustand gleichmäßig in den zweiten übergeht. Es wird verwendet, um ein Anzeigeobjekt über die Bühne zu bewegen und im Zeitverlauf verschiedene Änderungen daran vorzunehmen, wie vergrößern, verkleinern, drehen, ausblenden oder Farbe ändern.

Kopieren von Bewegungs-Tween-Skripts in Flash

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Ein Tween generiert Zwischenbilder, die ein Anzeigeobjekt in zwei verschiedenen Bildern der Zeitleiste in einem jeweils anderen Zustand zeigen. Es vermittelt den Eindruck, dass das erste Bild gleichmäßig in das zweite Bild übergeht. In der Regel ändert sich bei einem Bewegungs-Tween die Position des Anzeigeobjekts, wodurch eine Bewegung entsteht. Zusätzlich zum Ändern der Position des Anzeigeobjekts kann ein Bewegungs-Tween das Anzeigeobjekt aber auch drehen, neigen, vergrößern, verkleinern oder mit Filtern versehen.

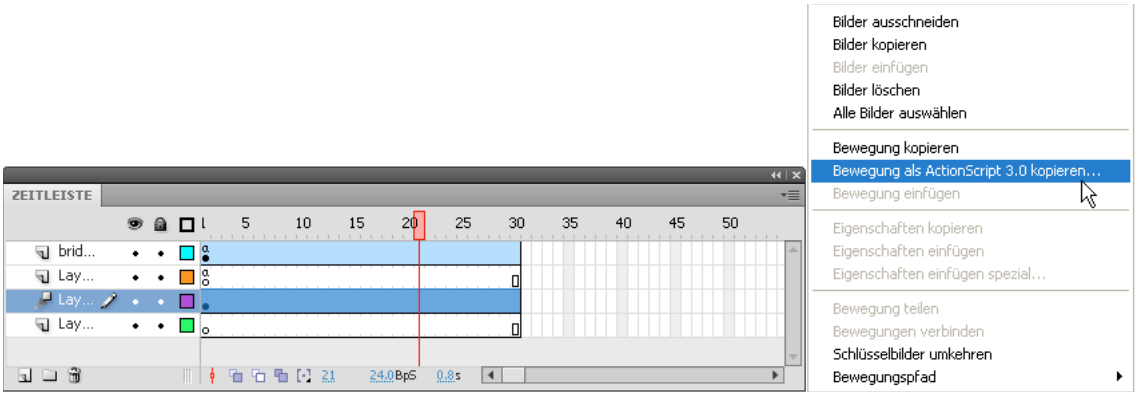
Sie erstellen ein Bewegungs-Tween in Flash, indem Sie ein Anzeigeobjekt zwischen Schlüsselbildern entlang der Zeitleiste bewegen. Der ActionScript-Code zur Beschreibung des Tweens wird in Flash automatisch erstellt. Sie können diesen Code kopieren und in einer Datei speichern. Weitere Informationen zum Erstellen eines Bewegungs-Tweens finden Sie im Abschnitt zu Bewegungs-Tweens in der Dokumentation *Verwenden von Flash Professional*.

Es gibt zwei Möglichkeiten, um in Flash auf den Befehl „Bewegung als ActionScript 3.0 kopieren“ zuzugreifen. Erstens können Sie das Kontextmenü eines Tweens auf der Bühne verwenden:

1 Wählen Sie das Bewegungs-Tween auf der Bühne aus.

2 Klicken Sie mit der rechten Maustaste (Windows) bzw. bei gedrückter Ctrl-Taste (Macintosh).

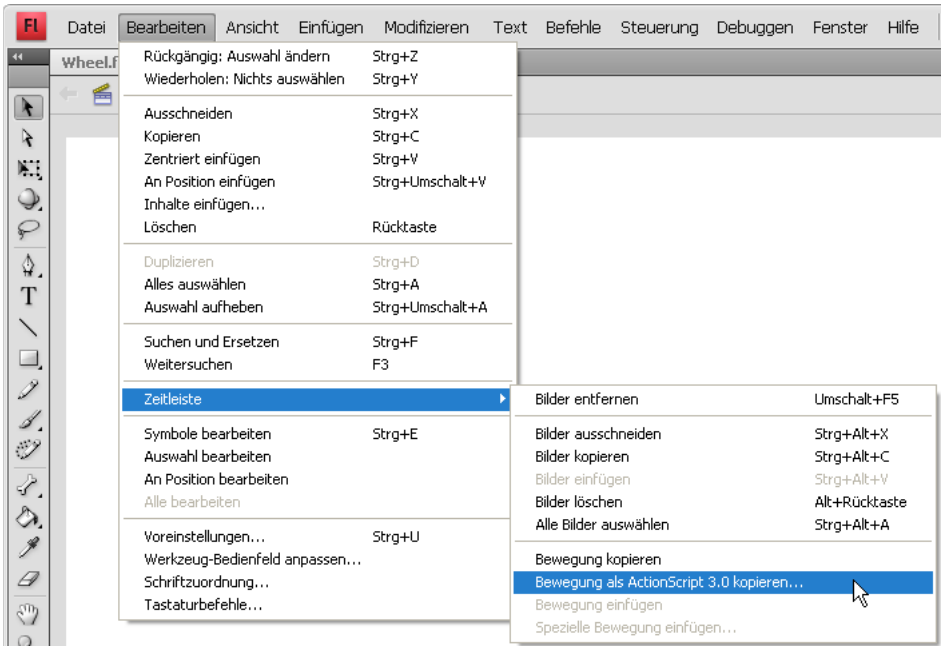
3 Wählen Sie „Bewegung als ActionScript 3.0 kopieren“.



Zweitens können Sie den Befehl direkt im Menü „Bearbeiten“ von Flash auswählen:

1 Wählen Sie das Bewegungs-Tween auf der Bühne aus.

2 Wählen Sie „Bearbeiten“ > „Zeitleiste“ > „Bewegung als ActionScript 3.0 kopieren“.



Nachdem Sie das Skript kopiert haben, fügen Sie es in eine Datei ein und speichern Sie die Datei.

Nachdem Sie ein Bewegungs-Tween erstellt und das Skript kopiert und gespeichert haben, können Sie es unverändert wieder verwenden oder in Ihren eigenen dynamischen ActionScript-Animationen ändern.

Einbinden von Bewegungs-Tween-Skripts

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

In der Kopfzeile des ActionScript-Codes, den Sie aus Flash kopieren, werden alle Module aufgelistet, die für das Bewegungs-Tween erforderlich sind.

Klassen für Bewegungs-Tweens

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Die wichtigsten Bewegungs-Tween-Klassen sind AnimatorFactory, MotionBase und Motion im `fl.motion`-Paket. Abhängig von den Eigenschaften, die vom Bewegungs-Tween bearbeitet werden, benötigen Sie unter Umständen weitere Klassen. Wenn das Bewegungs-Tween beispielsweise das Anzeigeobjekt transformiert oder dreht, müssen Sie die entsprechenden `flash.geom`-Klassen importieren. Wenn das Bewegungs-Tween Filter anwendet, importieren Sie die `flash.filter`-Klassen. In ActionScript handelt es sich bei einem Bewegungs-Tween um eine Instanz der Motion-Klasse. In der Motion-Klasse wird eine Schlüsselbildanimationssequenz gespeichert, die auf ein visuelles Objekt angewendet werden kann. Die Animationsdaten beinhalten Position, Skalierung, Drehung, Neigung, Farbe, Filter und Beschleunigung.

Der folgende ActionScript-Code wurde aus einem Bewegungs-Tween kopiert, das zur Animation eines Anzeigeobjekts mit dem Instanznamen `Symbol1_2` in Flash erstellt wurde. Der Code deklariert eine Variable für ein MotionBase-Objekt namens `__motion_Symbol1_2`. MotionBase ist die übergeordnete Klasse der Motion-Klasse.

```
var __motion_Symbol1_2:MotionBase;
```

Anschließend erstellt das Skript das Motion-Objekt:

```
__motion_Symbol1_2 = new Motion();
```

Motion-Objektnamen

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Im vorigen Beispiel hat Flash den Namen `__motion_Symbol1_2` automatisch für das Motion-Objekt generiert. Das Präfix `__motion_` wurde dem Namen des Anzeigeobjekts angefügt. Daher basiert der automatisch generierte Name auf dem Instanznamen des Zielobjekts des Bewegungs-Tweens in Flash. Die `duration`-Eigenschaft des Motion-Objekts gibt die Gesamtzahl der Bilder im Bewegungs-Tween an:

```
__motion_Symbol1_2.duration = 200;
```

Standardmäßig benennt Flash automatisch die Anzeigeobjektinstanz, deren Bewegungs-Tween kopiert wird, wenn nicht bereits ein Instanzname vorhanden ist.

Wenn Sie von Flash erstelltes ActionScript in Ihrer eigenen Animation wiederverwenden, können Sie den Namen, den Flash automatisch für das Tween generiert, beibehalten oder ihn durch einen anderen Namen ersetzen. Wenn Sie den Tween-Namen ändern, müssen Sie darauf achten, ihn im ganzen Skript zu ändern.

Stattdessen könnten Sie in Flash dem Zielobjekt des Bewegungs-Tweens auch einen selbst gewählten Namen zuweisen. Dann erstellen Sie das Bewegungs-Tween und kopieren das Skript. Unabhängig vom Benennungsverfahren müssen Sie darauf achten, dass jedes Motion-Objekt in Ihrem ActionScript-Code über einen eindeutigen Namen verfügt.

Beschreiben der Animation

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Die `addPropertyArray()`-Methode der MotionBase-Klasse fügt ein Array aus Werten hinzu, um jede getweente Eigenschaft zu beschreiben.

Möglicherweise enthält das Array ein Array-Element für jedes Schlüsselbild im Bewegungs-Tween. Häufig liegt die Gesamtanzahl der Elemente in diesen Arrays unter der Anzahl der Schlüsselbilder im Bewegungs-Tween. Dies ist der Fall, wenn der letzte Wert im Array sich für die verbleibenden Bilder nicht ändert.

Wenn die Länge des Array-Arguments größer als die `duration`-Eigenschaft des Motion-Objekts ist, passt `addPropertyArray()` den Wert der `duration`-Eigenschaft entsprechend an. Es fügt keine Schlüsselbilder für die zuvor hinzugefügten Eigenschaften hinzu. Die neu hinzugefügten Schlüsselbilder werden für die zusätzlichen Bilder der Animation beibehalten.

Die Eigenschaften `x` und `y` des Motion-Objekts beschreiben die Positionsänderung des getweenten Objekts, während die Animation ausgeführt wird. Bei diesen Koordinaten handelt es sich um die Werte, die sich mit größter Wahrscheinlichkeit in jedem Schlüsselbild ändern, wenn die Position des Anzeigeobjekts sich ändert. Sie können zusätzliche Bewegungseigenschaften über die Methode `addPropertyArray()` hinzufügen. Fügen Sie bei einer Größenänderung des getweenten Objekts beispielsweise die Werte `scaleX` and `scaleY` hinzu. Wenn das Objekt geneigt wird, fügen Sie die Werte `skewX` und `skewY` hinzu. Bei einer Drehung fügen Sie die Eigenschaft `rotationConcat` hinzu.

Verwenden Sie die Methode `addPropertyArray()`, um die folgenden Tween-Eigenschaften zu definieren:

`x`	Horizontale Position des Transformationspunkts des Objekts im Koordinatensystem des übergeordneten Objekts.
`y`	Vertikale Position des Transformationspunkts des Objekts im Koordinatensystem des übergeordneten Objekts.
`z`	Tiefenposition (Z-Achse) des Transformationspunkts des Objekts im Koordinatensystem des übergeordneten Objekts.
`scaleX`	Horizontale Skalierung des Objekts als Prozentwert, ausgehend vom Transformationspunkt.
`scaleY`	Vertikale Skalierung des Objekts als Prozentwert, ausgehend vom Transformationspunkt.
`skewX`	Horizontaler Neigungswinkel des Zielobjekts in Grad, ausgehend vom Transformationspunkt.
`skewY`	Vertikaler Neigungswinkel des Zielobjekts in Grad, ausgehend vom Transformationspunkt.
`rotationX`	Drehung des Objekts um die X-Achse, ausgehend von der ursprünglichen Ausrichtung.
`rotationY`	Drehung des Objekts um die Y-Achse, ausgehend von der ursprünglichen Ausrichtung.
`rotationConcat`	Drehungswerte (Z-Achse) des Objekts in der Bewegung relativ zur vorherigen Ausrichtung, ausgehend vom Transformationspunkt.
`useRotationConcat`	Wenn diese Eigenschaft festgelegt ist, wird das Zielobjekt gedreht, wenn `addPropertyArray()` Bewegungsdaten angibt.
`blendMode`	Wert der BlendMode-Klasse, der die Farbmischung des Objekts mit den darunterliegenden Grafiken angibt.
`matrix3D`	matrix3D-Eigenschaft, sofern für das Schlüsselbild vorhanden; wird für 3D-Tweens verwendet. Bei Verwendung dieser Eigenschaft werden alle vorigen Transformationseigenschaften ignoriert.
`rotationZ`	Drehung des Objekts um die Z-Achse in Grad, ausgehend von der ursprünglichen Ausrichtung relativ zum übergeordneten 3D-Container; wird für 3D-Tweens anstelle von rotationConcat verwendet.

Die Eigenschaften, die im automatisch generierten Skript hinzugefügt werden, richten sich nach den Eigenschaften, die dem Bewegungs-Tween in Flash zugewiesen wurden. Wenn Sie Ihre eigene Version des Skripts anpassen, können Sie einige dieser Eigenschaften hinzufügen, entfernen oder ändern.

Der folgende Code dient zum Zuweisen von Werten zu den Eigenschaften eines Bewegungs-Tweens namens `__motion_Wheel`. In diesem Fall ändert sich die Position des getweenten Anzeigeobjekts nicht, sondern in allen 29 Bildern des Bewegungs-Tweens dreht sich das Objekt an seiner festen Position. Die verschiedenen Werte, die dem `rotationConcat`-Array zugewiesen sind, definieren die Drehung. Die anderen Eigenschaftswerte dieses Bewegungs-Tweens ändern sich nicht.

```
__motion_Wheel = new Motion();
__motion_Wheel.duration = 29;
__motion_Wheel.addPropertyArray("x", [0]);
__motion_Wheel.addPropertyArray("y", [0]);
__motion_Wheel.addPropertyArray("scaleX", [1.00]);
__motion_Wheel.addPropertyArray("scaleY", [1.00]);
__motion_Wheel.addPropertyArray("skewX", [0]);
__motion_Wheel.addPropertyArray("skewY", [0]);
__motion_Wheel.addPropertyArray("rotationConcat",
    [
        0,-13.2143,-26.4285,-39.6428,-52.8571,-66.0714,-79.2857,-92.4999,-105.714,
        -118.929,-132.143,-145.357,-158.571,-171.786,-185,-198.214,-211.429,-224.643,
        -237.857,-251.071,-264.286,-277.5,-290.714,-303.929,-317.143,-330.357,
        -343.571,-356.786,-370
    ]
);
__motion_Wheel.addPropertyArray("blendMode", ["normal"]);
```

Im nächsten Beispiel bewegt sich das Anzeigeobjekt namens `Leaf_1` über die Bühne. Die Arrays für seine `x`- und `y`-Eigenschaften enthalten unterschiedliche Werte für jedes der 100 Bilder in der Animation. Außerdem dreht sich das Objekt um seine `z`-Achse, während es sich über die Bühne bewegt. Die Elemente im Array der `rotationZ`-Eigenschaft bestimmen die Drehung.

```
__motion_Leaf_1 = new MotionBase();
__motion_Leaf_1.duration = 100;
__motion_Symbol1_4.addPropertyArray("y",
    [
        0,5.91999,11.84,17.76,23.68,29.6,35.52,41.44,47.36,53.28,59.2,65.12,71.04,
        76.96,82.88,88.8,94.72,100.64,106.56,112.48,118.4,124.32,130.24,136.16,142.08,
        148,150.455,152.909,155.364,157.818,160.273,162.727,165.182,167.636,170.091,
        172.545,175,177.455,179.909,182.364,184.818,187.273,189.727,192.182,194.636,
        197.091,199.545,202,207.433,212.865,218.298,223.73,229.163,234.596,240.028,
        245.461,250.893,256.326,261.759,267.191,272.624,278.057,283.489,
        288.922,294.354,299.787,305.22,310.652,316.085,321.517,326.95,330.475,334,
        337.525,341.05,344.575,348.1,351.625,355.15,358.675,362.2,365.725,369.25,
        372.775,376.3,379.825,383.35,386.875,390.4,393.925,397.45,400.975,404.5,
        407.5,410.5,413.5,416.5,419.5,422.5,425.5
    ]
);
__motion_Symbol1_4.addPropertyArray("scaleX", [1.00]);
__motion_Symbol1_4.addPropertyArray("scaleY", [1.00]);
__motion_Symbol1_4.addPropertyArray("skewX", [0]);
__motion_Symbol1_4.addPropertyArray("skewY", [0]);
__motion_Symbol1_4.addPropertyArray("z",
    [
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ]
);
```

```
__motion_Symbol1_4.addPropertyArray("rotationX", [64.0361]);
__motion_Symbol1_4.addPropertyArray("rotationY", [41.9578]);
__motion_Symbol1_4.addPropertyArray("rotationZ",
    [
        -18.0336,-17.5536,-17.0736,-16.5936,-16.1136,-15.6336,-15.1536,-14.6736,
        -14.1936,-13.7136,-13.2336,-12.7536,-12.2736,-11.7936,-11.3136,-10.8336,
        -10.3536,-9.8736,-9.3936,-8.9136,-8.4336,-7.9536,-7.4736,-6.9936,-6.5136,
        -6.0336,-7.21542,-8.39723,-9.57905,-10.7609,-11.9427,-13.1245,-14.3063,
        -15.4881,-16.67,-17.8518,-19.0336,-20.2154,-21.3972,-22.5791,-23.7609,
        -24.9427,-26.1245,-27.3063,-28.4881,-29.67,-30.8518,-32.0336,-31.0771,
        -30.1206,-29.164,-28.2075,-27.251,-26.2945,-25.338,-24.3814,-23.4249,
        -22.4684,-21.5119,-20.5553,-19.5988,-18.6423,-17.6858,-16.7293,-15.7727
        -14.8162,-13.8597,-12.9032,-11.9466,-10.9901,-10.0336,-10.9427,-11.8518,
        -12.7609,-13.67,-14.5791,-15.4881,-16.3972,-17.3063,-18.2154,-19.1245,
        -20.0336,-20.9427,-21.8518,-22.7609,-23.67,-24.5791,-25.4881,-26.3972,
        -27.3063,-28.2154,-29.1245,-30.0336,-28.3193,-26.605,-24.8907,-23.1765,
        -21.4622,-19.7479,-18.0336
    ]
);
__motion_Symbol1_4.addPropertyArray("blendMode", ["normal"]);
```

Hinzufügen von Filtern

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Wenn das Zielobjekt eines Bewegungs-Tweens Filter enthält, werden diese Filter mit den Methoden `initFilters()` und `addFilterPropertyArray()` der Motion-Klasse hinzugefügt.

Initialisieren des Filter-Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Die Filter werden mit der Methode `initFilters()` initialisiert. Das erste Argument ist ein Array der vollständig qualifizierten Klassennamen aller auf das Anzeigeobjekt angewendeten Filter. Dieses Array aus Filternamen wird anhand der Filterliste für das Bewegungs-Tween in Flash generiert. In Ihrer Kopie des Skripts können Sie für dieses Array alle Filter des `flash.filters`-Pakets hinzufügen oder entfernen. Mit dem folgenden Aufruf wird die Filterliste für das Ziel-Anzeigeobjekt initialisiert. Der Code wendet die Filter `DropShadowFilter`, `GlowFilter` und `BevelFilter` an und kopiert die Liste in jedes Schlüsselbild im Motion-Objekt.

```
__motion_Box.initFilters(["flash.filters.DropShadowFilter", "flash.filters.GlowFilter",
"flash.filters.BevelFilter"], [0, 0, 0]);
```

Hinzufügen von Filtern

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Die Methode `addFilterPropertyArray()` beschreibt die Eigenschaften eines initialisierten Filters mit den folgenden Argumenten:

1. Das erste Argument identifiziert einen Filter nach der Indexposition. Die Indexposition verweist auf die Position des Filternamens im Array der Filterklassennamen, das in einem früheren Aufruf von `initFilters()` übergeben wurde.

2 Beim zweiten Argument handelt es sich um die Filtereigenschaft, die für diesen Filter in jedem Schlüsselbild gespeichert werden soll.

3 Das dritte Argument ist der Wert der angegebenen Filtereigenschaft.

Ausgehend vom vorigen `initFilters()`-Aufruf weisen die folgenden `addFilterPropertyArray()`-Aufrufe den Eigenschaften `blurX` und `blurY` von `DropShadowFilter` den Wert 5 zu. `DropShadowFilter` ist das erste Element (Indexposition 0) im Array der initialisierten Filter:

```
__motion_Box.addFilterPropertyArray(0, "blurX", [5]);
__motion_Box.addFilterPropertyArray(0, "blurY", [5]);
```

Mit den nächsten drei Aufrufen werden den quality-, alpha- und color-Eigenschaften von `GlowFilter` Werte zugewiesen. Dieser Filter ist das zweite Element (Indexposition 1) im Array der initialisierten Filter:

```
__motion_Box.addFilterPropertyArray(1, "quality", [BitmapFilterQuality.LOW]);
__motion_Box.addFilterPropertyArray(1, "alpha", [1.00]);
__motion_Box.addFilterPropertyArray(1, "color", [0xff0000]);
```

Die nächsten vier Aufrufe weisen den Eigenschaften `shadowAlpha`, `shadowColor`, `highlightAlpha` und `highlightColor` von `BevelFilter` Werte zu. Dabei handelt es sich um das dritte Element (Indexposition 2) im Array der initialisierten Filter:

```
__motion_Box.addFilterPropertyArray(2, "shadowAlpha", [1.00]);
__motion_Box.addFilterPropertyArray(2, "shadowColor", [0x000000]);
__motion_Box.addFilterPropertyArray(2, "highlightAlpha", [1.00]);
__motion_Box.addFilterPropertyArray(2, "highlightColor", [0xffffff]);
```

Anpassen der Farbe mit ColorMatrixFilter

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Nach der Initialisierung von `ColorMatrixFilter` können Sie die entsprechenden `AdjustColor`-Eigenschaften festlegen, um Helligkeit, Kontrast, Sättigung und Farbton des getweenten Anzeigeobjekts einzustellen. Normalerweise wird der `AdjustColor`-Filter angewendet, wenn das Bewegungs-Tween in Flash erstellt wird; die Feinabstimmung können Sie in Ihrer Kopie des ActionScript vornehmen. Mit dem folgenden Codebeispiel werden Farbton und Sättigung des Anzeigeobjekts während der Bewegung transformiert.

```
__motion_Leaf_1.initFilters(["flash.filters.ColorMatrix"], [0], -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorBrightness", [0], -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorContrast", [0], -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorSaturation",
    [
        0,-0.589039,1.17808,-1.76712,-2.35616,-2.9452,-3.53424,-4.12328,
        -4.71232,-5.30136,-5.89041, 6.47945,-7.06849,-7.65753,-8.24657,
        -8.83561,-9.42465,-10.0137,-10.6027,-11.1918,11.7808,-12.3699,
        -12.9589,-13.5479,-14.137,-14.726,-15.3151,-15.9041,-16.4931,
        17.0822,-17.6712,-18.2603,-18.8493,-19.4383,-20.0274,-20.6164,
        -21.2055,-21.7945,22.3836,-22.9726,-23.5616,-24.1507,-24.7397,
        -25.3288,-25.9178,-26.5068,-27.0959,27.6849,-28.274,-28.863,-29.452,
        -30.0411,-30.6301,-31.2192,-31.8082,-32.3973,32.9863,-33.5753,
        -34.1644,-34.7534,-35.3425,-35.9315,-36.5205,-37.1096,-37.6986,
        38.2877,-38.8767,-39.4657,-40.0548,-40.6438,-41.2329,-41.8219,
        -42.411,-43
    ],
    -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorHue",
    [
        0,0.677418,1.35484,2.03226,2.70967,3.38709,4.06451,4.74193,5.41935,
        6.09677,6.77419,7.45161,8.12903,8.80645,9.48387,10.1613,10.8387,11.5161,
        12.1935,12.871,13.5484,14.2258,14.9032,15.5806,16.2581,16.9355,17.6129,
        18.2903,18.9677,19.6452,20.3226,21,22.4286,23.8571,25.2857,26.7143,28.1429,
        29.5714,31,32.4286,33.8571,35.2857,36.7143,38.1429,39.5714,41,42.4286,43.8571,
        45.2857,46.7143,48.1429,49.5714,51,54,57,60,63,66,69,72,75,78,81,84,87,
        90,93,96,99,102,105,108,111,114
    ],
    -1, -1);
```

Verknüpfen von Bewegungs-Tweens mit den zugehörigen Anzeigeobjekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher, erfordert Flash CS3 oder höher

Die letzte Aufgabe besteht darin, das Bewegungs-Tween mit den Anzeigeobjekten zu verknüpfen, die es beeinflusst.

Die AnimatorFactory-Klasse verwaltet die Beziehung zwischen einem Bewegungs-Tween und seinen Zielanzeigeobjekten. Als Argument für den AnimatorFactory-Konstruktur dient das Motion-Objekt:

```
var __animFactory_Wheel:AnimatorFactory = new AnimatorFactory(__motion_Wheel);
```

Verwenden Sie die `addTarget()`-Methode der AnimatorFactory-Klasse, um das Zielanzeigeobjekt mit seinem Bewegungs-Tween zu verknüpfen. Im aus Flash kopierten ActionScript-Code ist die Zeile `addTarget()` auskommentiert und es wird kein Instanzname angegeben:

```
// __animFactory_Wheel.addTarget(<instance name goes here>, 0);
```

Geben Sie in Ihrer Kopie das Anzeigeobjekt an, das mit dem Bewegungs-Tween verknüpft werden soll. Im folgenden Beispiel werden die Ziele als `greenWheel` und `redWheel` angegeben:

```
__animFactory_Wheel.AnimatorFactory.addTarget(greenWheel, 0);
__animFactory_Wheel.AnimationFactory.addTarget(redWheel, 0);
```

Sie können mehrere Anzeigeobjekte mit dem Bewegungs-Tween verknüpfen, indem Sie `addTarget()` mehrmals aufrufen.

Kapitel 18: Arbeiten mit inverser Kinematik

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

Die inverse Kinematik (IK) ist eine leistungsstarke Technik zur Erstellung realistischer Bewegungen.

Mithilfe von IK können Sie innerhalb einer Kette verbundener Teile, eines sogenannten IK-Skeletts, koordinierte Bewegungen erstellen, damit die Teile sich realitätsgetreu zusammen bewegen. Das Skelett setzt sich aus Knochen (Bones) und Gelenken zusammen. Ausgehend vom Endpunkt des Skeletts berechnet IK die Winkel der Gelenke, die erforderlich sind, damit der Endpunkt erreicht wird.

Eine manuelle Berechnung dieser Winkel wäre äußerst schwierig. Der große Vorteil dieser Funktion liegt darin, dass Sie Skelette interaktiv mit Adobe® Flash® Professional erstellen können. Anschließend animieren Sie die Skelette mit ActionScript. Die in Flash Professional enthaltene IK-Engine führt die Berechnungen aus, die die Bewegungen des Skeletts beschreiben. Sie können die Bewegung im ActionScript-Code auf bestimmte Parameter beschränken.

Neu in der IK-Version von Flash Professional CS5 ist das Konzept der Bone-Federung, das bisher hauptsächlich von Spezialanwendungen für die Animation angeboten wurde. In Kombination mit dem neuen dynamischen Physikmodul (Physics Engine) ermöglicht dieses Merkmal die Konfiguration von realistischen Bewegungen. Dieser Effekt ist sowohl zur Laufzeit als auch beim Authoring sichtbar.

Zur Erstellung von IK-Skeletten benötigen Sie eine Lizenz für Flash Professional.

Verwandte Hilfethemen

fl.ik-Paket

Grundlagen der inversen Kinematik

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

Mit der inversen Kinematik (IK) können Sie realitätsgetreue Animationen erstellen, indem Sie Teile so verbinden, dass sie sich zusammen auf realistische Weise bewegen.

Beispielsweise können Sie mithilfe von IK ein Bein an eine bestimmte Position bewegen, indem Sie die dazu erforderlichen Beingelenkbewegungen nachbilden. IK stützt sich auf eine miteinander verbundene Bone-Struktur, die als IK-Skelett bezeichnet wird. Verwenden Sie das `fl.ik`-Paket zur Erstellung von Animationen mit realitätsgetreuen Bewegungsabläufen. Mithilfe dieses Pakets können Sie mehrere IK-Skelette nahtlos animieren, ohne dass Sie die Physik der IK-Algorithmen genau verstehen müssen.

Sie erstellen das IK-Skelett mit seinen zugehörigen Bones und Gelenken in Flash Professional. Anschließend können Sie auf die IK-Klassen zugreifen, um das Skelett zur Laufzeit zu animieren.

Ausführliche Anleitungen zur Erstellung eines IK-Skeletts finden Sie im Abschnitt zur inversen Kinematik im Handbuch *Verwenden von Flash Professional*.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die mit dieser Funktion zu tun haben:

Skelett Eine kinematische Kette aus Bones (Knochen) und Gelenken, die bei der Computeranimation zur Simulation von realistischen Bewegungsabläufen verwendet wird.

Bone Ein starres Segment in einem Skelett, das sich mit einem Knochen eines Tierskeletts vergleichen lässt.

Inverse Kinematik (IK) Verfahren zum Bestimmen der Parameter für ein verbundenes flexibles Objekt, das als kinematische Kette oder Skelett bezeichnet wird.

Gelenk Die Kontaktstelle zweier Knochen (Bones), die die Bewegung der Knochen ermöglicht; analog zum Gelenk eines Tieres.

Physikmodul (Physics Engine) Ein Paket mit physikalischen Algorithmen, mit denen in Animationen realistische Bewegungen und Aktionen erstellt werden können.

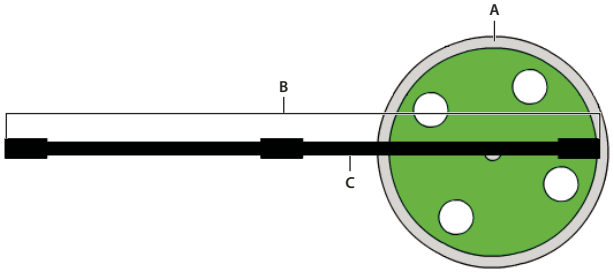
Sprungfeder Die Bewegung eines Bone als Reaktion auf eine Bewegung des übergeordneten Bone, die sich im Zeitverlauf schrittweise abschwächt.

Animieren von IK-Skeletten – Überblick

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

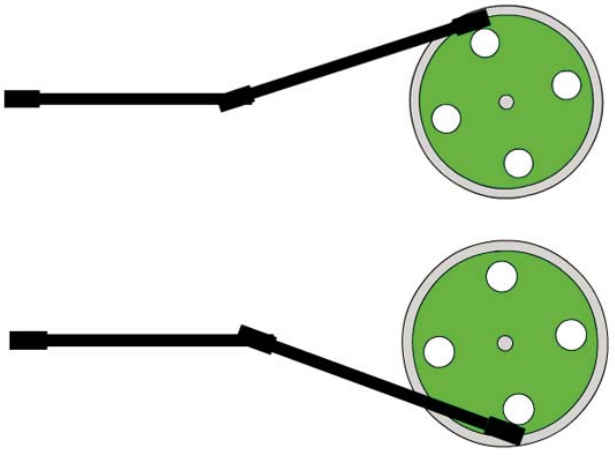
Nach der Erstellung eines IK-Skeletts in Flash Professional verwenden Sie die `fl.ik`-Klassen, um die Bewegung einzuschränken, die Ereignisse zu verfolgen und das Skelett zur Laufzeit zu animieren.

Die folgende Abbildung zeigt einen Movieclip namens `Wheel`. Bei der Achse handelt es sich um eine Instanz eines IK-Skeletts namens `Axle`. Die IKMover-Klasse bewegt das Skelett synchron mit der Drehung des Rades. Der IK-Bone `ikBone2` des Skeletts ist am Rückgelenk mit dem Rad verbunden.



A. *Rad* ***B.*** *Achse* ***C.*** *ikBone2*

Zur Laufzeit dreht sich das Rad gemäß dem Bewegungs-Tween `__motion_Wheel`, das unter „Beschreiben der Animation“ auf Seite 361 erläutert wird Ein IKMover-Objekt initialisiert und steuert die Bewegung der Achse. Die folgende Abbildung zeigt zwei Schnappschüsse des Achsenskeletts, das jeweils an unterschiedlichen Positionen am sich drehenden Rad angefügt ist.



Zur Laufzeit führt der folgende ActionScript-Code diese Aufgaben aus:

- Abrufen von Informationen über das Skelett und seine Komponenten
- Instanziieren eines IKMover-Objekts
- Bewegen der Achse zusammen mit der Drehung des Rads

```
import fl.ik.*

var tree:IKArmature = IKManager.getArmatureByName("Axle");
var bone:IKBone = tree.getBoneByName("ikBone2");
var endEffector:IKJoint = bone.tailJoint;
var pos:Point = endEffector.position;

var ik:IKMover = new IKMover(endEffector, pos);
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;

Wheel.addEventListener(Event.ENTER_FRAME, frameFunc);

function frameFunc(event:Event)
{
    if (Wheel != null)
    {
        var mat:Matrix = Wheel.transform.matrix;
        var pt = new Point(90, 0);
        pt = mat.transformPoint(pt);

        ik.moveTo(pt);
    }
}
```

Zur Bewegung der Achse werden die folgenden IK-Klassen verwendet:

- IKArmature: beschreibt das aus Bones (Knochen) und Gelenken bestehende Skelett; muss in Flash Professional erstellt werden.
- IKManager: Container-Klasse für alle IK-Skelette im Dokument; muss in Flash Professional erstellt werden.
- IKBone: ein Segment eines IK-Skeletts.
- IKJoint: eine Verbindung zwischen zwei IK-Bones.
- IKMover: initialisiert und steuert die Bewegungen von IK-Skeletten.

Ausführliche Erläuterungen dieser Klassen finden Sie in der Beschreibung zum ik-Paket.

Abrufen von Informationen über IK-Skelette

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

Zunächst deklarieren Sie Variablen für das Skelett, den Bone und das Gelenk, aus denen die zu bewegenden Strukturen bestehen.

Im folgenden Code wird die `getArmatureByName()`-Methode der IKManager-Klasse verwendet, um den Wert der Axle-Struktur der IKArmature-Variablen `tree` zuzuweisen. Das Axle-Skelett wurde zuvor mit Flash Professional erstellt.

```
var tree:IKArmature = IKManager.getArmatureByName("Axle");
```

Ähnlich wird im folgenden Code die `getBoneByName()`-Methode der IKArmature-Klasse verwendet, um der IKBone-Variablen den Wert des Bones `ikBone2` zuzuweisen.

```
var bone:IKBone = tree.getBoneByName("ikBone2");
```

Das Rückgelenk des Bones `ikBone2` ist der Teil des Skeletts, der an das Drehrad angefügt wird.

Die folgende Codezeile deklariert die Variable `endEffector` und weist ihr die `tailjoint`-Eigenschaft des Bones `ikBone2` zu:

```
var endEffector:IKJoint = home.tailjoint;
```

In der Variablen `pos` wird die aktuelle Position des `endEffector`-Gelenks gespeichert.

```
var pos:Point = endEffector.position;
```

In diesem Beispiel ist `pos` die Position des Gelenks am Ende der Achse und am Verbindungspunkt mit dem Rad. Der ursprüngliche Wert dieser Variablen stammt aus der `position`-Eigenschaft von IKJoint.

Instanziieren eines IKMover-Objekts und Einschränken seiner Bewegung

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

Die Achse wird von einer Instanz der IKMover-Klasse bewegt.

Mit der folgenden Codezeile wird das IKMover-Objekt `ik` instanziiert. Dabei werden das zu bewegende Element und der Startpunkt der Bewegung an den Konstruktor übergeben:

```
var ik:IKMover = new IKMover(endEffector, pos);
```

Mit den Eigenschaften der IKMover-Klasse können Sie die Bewegung eines Skeletts einschränken. Die Bewegung lässt sich nach Entfernung, Iterationen und Zeit der Bewegung einschränken.

Diese Einschränkungen werden mit den folgenden Eigenschaftspaaren erzwungen. Die Paare bestehen aus einem booleschen Wert, der angibt, ob die Bewegung eingeschränkt ist, und aus einer Ganzzahl, die die Einschränkung definiert:

Boolean-Eigenschaft	Integer-Eigenschaft	Festgelegte Beschränkung
`limitByDistance:Boolean`	`distanceLimit:int`	Legt die maximale Entfernung der Bewegung in Pixel fest, die das IK-Modul bei jeder Iteration durchführt.
`limitByIteration:Boolean`	`iterationLimit:int`	Legt die maximale Anzahl der Iterationen fest, die das IK-Modul für jede Bewegung durchführt.
`limitByTime:Boolean`	`timeLimit:int`	Legt die maximale Zeit in Millisekunden fest, die dem IK-Modul zur Durchführung der Bewegung zur Verfügung steht.

Standardmäßig sind alle Boolean-Eigenschaften auf `false` eingestellt, die Bewegung ist also nicht eingeschränkt, sofern Sie nicht explizit einen Boolean-Wert auf `true` festlegen. Um eine Beschränkung festzulegen, setzen Sie die jeweilige Eigenschaft auf `true` und geben Sie einen Wert für die entsprechende Integer-Eigenschaft an. Wenn Sie die Einschränkung auf einen Wert festlegen, ohne jedoch die entsprechende Boolean-Eigenschaft festzulegen, wird die Einschränkung ignoriert. In diesem Fall wird das Objekt vom IK-Modul weiter bewegt, bis eine andere Einschränkung oder die Zielposition des IKMover-Objekts erreicht wird.

Im folgenden Beispiel ist die maximale Entfernung der Skelettbewegung auf 0,1 Pixel pro Iteration eingestellt. Die maximale Anzahl an Iterationen für jede Bewegung beträgt 10.

```
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;
```

Bewegen eines IK-Skeletts

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

IKMover bewegt die Achse innerhalb des Ereignis-Listeners für das Rad. Bei jedem enterFrame-Ereignis des Rades wird eine neue Zielposition für das Skelett berechnet. Unter Verwendung seiner `moveTo()`-Methode bewegt das IKMover-Objekt das Rückgelenk an die Zielposition oder so weit wie innerhalb der Einschränkungen möglich, die von den Eigenschaften `limitByDistance`, `limitByIteration` und `limitByTime` vorgegeben sind.

```
Wheel.addEventListener(Event.ENTER_FRAME, frameFunc);

function frameFunc(event:Event)
{
    if (Wheel != null)
    {
        var mat:Matrix = Wheel.transform.matrix;
        var pt = new Point(90,0);
        pt = mat.transformPoint(pt);

        ik.moveTo(pt);
    }
}
```

Verwenden von Federn

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS5 oder höher

Die inverse Kinematik in Flash Professional CS5 unterstützt die Bone-Federung. Die Bone-Federung kann beim Authoring festgelegt werden. Attribute für Bone-Federn können zur Laufzeit hinzugefügt oder geändert werden. Die Spring-Eigenschaft gilt für einen Bone und seine Gelenke. Sie hat zwei Attribute: `IKJoint.springStrength` definiert die Stärke der Federung, während `IKJoint.springDamping` der Federungsstärke einen Widerstand entgegensetzt und sich auf die Abschwächung der Federung auswirkt.

Die Federungsstärke hat einen Wert von 0-100 %. Beim Standardwert 0 wird eine völlig starre Feder verwendet, während beim Wert 100 eine sehr lose Feder angewendet wird, die durch physikalische Eigenschaften gesteuert wird. Bones mit Federn reagieren auf die Bewegung ihrer Gelenke. Wenn keine andere Versetzung (Drehung in x- oder y-Richtung) aktiviert ist, haben die Federeinstellungen keine Auswirkungen.

Die Federdämpfung hat einen Wert von 0-100 %. Beim Standardwert 0 wird kein Widerstand angewendet, während beim Wert 100 eine starke Dämpfung auf die Feder wirkt. Die Dämpfung wirkt sich darauf aus, wie lange es dauert, bis ein Bone nach dem Bewegungsanfang zur Ruheposition zurückkehrt.

Anhand der `IKArmature.springsEnabled`-Eigenschaft können Sie überprüfen, ob Federn für ein IKArmature-Objekt aktiviert sind. Die anderen Federeigenschaften und -methoden gehören zu individuellen IKJoint-Objekten. Ein Gelenk kann für eine Winkeldrehung und für das Versetzen entlang der x- und y-Achsen aktiviert werden. Der Drehungswinkel für eine Gelenkfeder kann über `IKJoint.setSpringAngle` festgelegt werden, die Versatzposition über `IKJoint.setSpringPt`.

In diesem Beispiel wird ein Bone anhand seines Namens ausgewählt und die tailJoint-Eigenschaft wird identifiziert. Der Code testet das übergeordnete Skelett, um festzustellen, ob Federn aktiviert sind. Dann werden die Federeigenschaften für das Gelenk festgelegt.

```
var arm:IKArmature = IKManager.getArmatureAt(0);
var bone:IKBone = arm.getBoneByName("c");
var joint:IKJoint = bone.tailJoint;
if (arm.springsEnabled) {
    joint.springStrength = 50; //medium spring strength
    joint.springDamping = 10; //light damping resistance
    if (joint.hasSpringAngle) {
        joint.setSpringAngle(30); //set angle for rotational spring
    }
}
```

Verwenden von IK-Ereignissen

Flash Player 10 und höher, Adobe AIR 1.5 und höher, erfordert Flash CS4 oder höher

Mithilfe der IKEvent-Klasse können Sie ein Ereignisobjekt erstellen, das Informationen über IK-Ereignisse enthält. IKEvent-Informationen beschreiben Bewegungen, die beendet wurden, da die angegebene Einschränkung der Zeit, Entfernung oder Iterationsanzahl erreicht wurden.

Das folgende Codebeispiel zeigt einen Ereignis-Listener und eine Ereignisprozedur zur Verfolgung von Ereignissen, die sich auf eine zeitliche Einschränkung beziehen. Diese Ereignisprozedur liefert Informationen zu Zeit, Entfernung, Iterationsanzahl und Gelenkeigenschaften eines Ereignisses, das ausgelöst wird, wenn das Zeitlimit des IKMover-Objekts überschritten wird.

```
var ikmover:IKMover = new IKMover(endjoint, pos);
ikMover.limitByTime = true;
ikMover.timeLimit = 1000;

ikmover.addEventListener(IKEvent.TIME_LIMIT, timeLimitFunction);

function timeLimitFunction(evt:IKEvent):void
{
    trace("timeLimit hit");
    trace("time is " + evt.time);
    trace("distance is " + evt.distance);
    trace("iterationCount is " + evt.iterationCount);
    trace("IKJoint is " + evt.joint.name);
}
```

Kapitel 19: Arbeiten mit drei Dimensionen (3D)

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Laufzeitumgebungen Flash Player und AIR unterstützen 3D-Grafiken auf zweierlei Weise. Sie können dreidimensionale Anzeigeobjekte in der Flash-Anzeigeliste verwenden. Diese Methode eignet sich zum Hinzufügen von dreidimensionalen Effekten zu Flash-Inhalten und für Objekte mit wenigen Polygonen. In Flash Player 11 und AIR 3 oder höher können Sie komplexe 3D-Szenen mit der Stage3D-API rendern.

Ein Stage3D-Viewport ist kein Anzeigeobjekt. Stattdessen werden die 3D-Grafiken in einen Viewport gerendert, der hinter der Flash-Anzeigeliste (und vor allen StageVideo-Viewportebenen) angezeigt wird. Anstatt eine Szene mit den Flash-DisplayObject-Klassen zu erstellen, verwenden Sie eine programmierbare 3D-Pipeline (ähnlich wie OpenGL und Direct3D). Diese Pipeline nimmt Dreieckdaten und Texturen als Eingabe und rendert die Szene mit den Shaderprogrammen, die Sie bereitstellen. Hardwarebeschleunigung wird verwendet, wenn auf dem Clientcomputer eine kompatible GPU mit unterstützten Treibern verfügbar ist.

Stage3D stellt eine API auf sehr niedriger Ebene bereit. In einer Anwendung sollten Sie ein 3D-Framework verwenden, dass Stage3D unterstützt. Sie können Ihr eigenes Framework erstellen, es sind aber auch verschiedene kommerzielle und Open-Source-Frameworks erhältlich.

Weitere Informationen zur Entwicklung von 3D-Anwendungen mit Stage3D und zu verfügbaren 3D-Frameworks finden Sie im Flash Player Developer Center: Stage 3D.

Grundlagen von 3D-Anzeigeobjekten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Der Hauptunterschied zwischen einem zweidimensionalen (2D) und einem dreidimensionalen (3D) Objekt, das auf einen zweidimensionalen Bildschirm projiziert wird, ist die Hinzufügung einer dritten Dimension zum Objekt. Durch die dritte Dimension kann das Objekt zum Standpunkt des Benutzers hin- und davon wegbewegt werden.

Wenn Sie die `z`-Eigenschaft eines Anzeigeobjekts ausdrücklich auf einen numerischen Wert setzen, erzeugt das Objekt automatisch eine 3D-Transformationsmatrix. Sie können diese Matrix verändern, indem Sie die 3D-Transformationseinstellungen dieses Objekts ändern.

Außerdem unterscheidet sich eine 3D-Drehung von einer 2D-Drehung. Bei 2D-Drehungen verläuft die Drehachse immer senkrecht zur x/y-Fläche, anders ausgedrückt: sie ist identisch mit der z-Achse. In einem 3D-Raum hingegen kann ein Objekt wahlweise um die x-, y- oder z-Achse gedreht werden. Durch das Festlegen der Dreh- und Skalierungseigenschaften eines Anzeigeobjekts lässt es sich im 3D-Raum bewegen.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die Ihnen beim Programmieren von dreidimensionalen Grafiken häufig begegnen:

Perspektive Die Abbildung paralleler Linien auf einer 2D-Fläche, wobei diese im Fluchtpunkt zusammenlaufen und somit den Eindruck von Tiefe und Entfernung schaffen.

Projektion Die Konstruktion von einem Objekt höherer Dimension auf einem 2D-Bild. Eine 3D-Projektion bildet die Punkte eines dreidimensionalen Körpers auf einer 2D-Fläche ab.

Drehen Ändern der Orientierung (und oft auch Position) eines Objekts durch kreisförmiges Bewegen eines jedes Punkts in einem Objekt um das Drehzentrum.

Transformation Ändern der 3D-Punkte oder Punktmenge durch Versetzen, Drehen, Skalieren (Vergrößern/Verkleinern), Neigen oder eine Kombination dieser Aktionen.

Versetzung Ändern der Position eines Objekts durch Verschieben eines jedes Punkts im Objekt um dieselbe Strecke und in dieselbe Richtung.

Fluchtpunkt Der Punkt in einer linearen, räumlichen Darstellung, an dem sich fliehende, parallele Linien zu treffen scheinen.

Vektor Ein 3D-Vektor repräsentiert einen Punkt oder eine Position im dreidimensionalen Raum bei Verwendung der kartesischen Koordinaten x, y und z.

Vertex Ein Eckpunkt.

Strukturiertes Gitter Jeder Punkt, der ein Objekt im 3D-Raum definiert.

UV-Zuordnung Eine Möglichkeit, 3D-Oberflächen mit Texturen oder Bitmaps zu versehen. Die UV-Zuordnung weist den Koordinaten auf einem Bild prozentuale Werte in Bezug auf die Horizontalachse (U-Achse) und Vertikalachse (V-Achse) zu.

T-Wert Der Skalierungsfaktor zum Bestimmen der Größe eines 3D-Objekts, während sich das Objekt zum aktuellen Standpunkt hin- bzw. davon wegbewegt.

Culling Ausschließen nicht sichtbarer Objektteile aus dem Rendervorgang. Mithilfe des Cullingverfahrens können Sie Oberflächen „auslesen“ und verbergen, die vom aktuellen Standpunkt aus nicht zu sehen sind, und somit die Rendergeschwindigkeit verbessern.

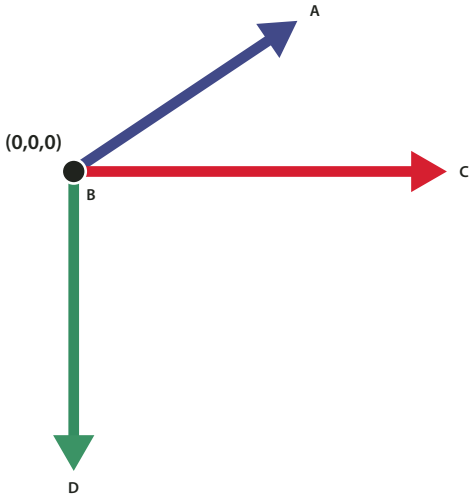
Informationen zu 3D-Anzeigeobjekten in Flash Player und der AIR-Laufzeitumgebung

Flash Player 10 und höher, Adobe AIR 1.5 und höher

In Flash Player-Versionen vor Flash Player 10 und Adobe AIR-Versionen vor Adobe AIR 1.5 verfügen Anzeigeobjekte über zwei Eigenschaften, `x` und `y`, mit denen sie auf einer 2D-Ebene positioniert werden. Ab Flash Player 10 und Adobe AIR 1.5 besitzt jedes ActionScript-Anzeigeobjekt auch eine `z`-Eigenschaft, die Ihnen die Positionierung entlang einer z-Achse erlaubt, die in der Regel zur Darstellung von Tiefe und Entfernung verwendet wird.

In Flash Player 10 und Adobe AIR 1.5 wurde die Unterstützung von 3D-Effekten eingeführt. Im Prinzip jedoch sind die Anzeigeobjekte flach. Jedes Anzeigeobjekt, z. B. ein MovieClip- oder ein Sprite-Objekt, wird letztendlich in zwei Dimensionen, auf einer einzigen Ebene, dargestellt. Die 3D-Funktionen ermöglichen Ihnen jedoch, diese Flächenobjekte in allen drei Dimensionen zu platzieren, zu verschieben, zu drehen oder in anderer Weise zu transformieren. Sie erlauben die Verarbeitung von 3D-Punkten und deren Konvertierung in zweidimensionale x/y-Koordinaten, sodass Sie 3D-Objekte auf eine 2D-Sicht projizieren können. Anhand dieser Funktion können Sie viele Arten der räumlichen Wahrnehmung simulieren.

Das von ActionScript verwendete dreidimensionales Koordinatensystem unterscheidet sich von anderen Koordinatensystemen. Wenn Sie in ActionScript zweidimensionale Koordinaten verwenden, erhöht sich der x-Wert beim Verschieben nach rechts entlang der x-Achse und der y-Wert erhöht sich beim Verschieben nach unten entlang der y-Achse. Das dreidimensionale Koordinatensystem behält diese Konventionen bei und fügt die z-Achse hinzu, deren Wert erhöht wird, wenn vom Standpunkt weg, also nach hinten, verschoben wird.



Die positiven Richtungen auf der x-, y- und z-Achse im dreidimensionalen ActionScript-Koordinatensystem
A. *Positive z-Achse* ***B.*** *Ursprung* ***C.*** *Positive x-Achse* ***D.*** *Positive y-Achse*

Hinweis: *Bedenken Sie, dass Flash Player und AIR dreidimensionale Anzeigeobjekte in Ebenen darstellt. Das bedeutet, dass Objekt A, wenn es in der Anzeigeliste vor Objekt B kommt, in Flash Player oder AIR immer vor Objekt B angezeigt wird, und zwar ungeachtet der z-Achsenwerte der beiden Objekte. Um diese Unstimmigkeit zwischen Anzeigeliste und z-Achsenreihenfolge aufzuheben, verwenden Sie die* `transform.getRelativeMatrix3D()`*-Methode, um die Ebenen von dreidimensionalen Anzeigeobjekten zu speichern und neu anzuordnen. Weitere Informationen finden Sie unter* *„Verwenden von 3DMatrix-Objekten zur Neuanordnung der Anzeige“ auf Seite 387.*

Die folgenden ActionScript-Klassen unterstützen die neuen 3D-bezogenen Funktionen:

1. Die flash.display.DisplayObject-Klasse verfügt über die `z`-Eigenschaft sowie über neue Dreh- und Skalierungseigenschaften für die Bearbeitung von Anzeigeobjekten im 3D-Raum. Die `DisplayObject.local3DToGlobal()`-Methode bietet eine einfache Möglichkeit, um 3D-Geometrie auf eine 2D-Fläche zu projizieren.
2. Die flash.geom.Vector3D-Klasse kann als Datenstruktur für die Verwaltung von 3D-Punkten verwendet werden. Sie unterstützt zudem die Vektorrechnung.
3. Die flash.geom.Matrix3D-Klasse unterstützt komplexe Transformationen von dreidimensionalen Körpern, wie Drehung, Skalierung und Verschiebung.
4. Die flash.geom.PerspectiveProjection-Klasse steuert die Parameter zur Abbildung von dreidimensionalen Körpern auf einer Fläche.

Es gibt zwei verschiedene Ansätze, um 3D-Bilder in ActionScript zu simulieren:

1 Das Anordnen und Animieren flacher Objekte im 3D-Raum. Dieser Ansatz beinhaltet die Animation von Anzeigeobjekten mithilfe ihrer x-, y- und z-Eigenschaften oder mithilfe der Dreh- und Skalierungseigenschaften der DisplayObject-Klasse. Komplexere Bewegungen können mithilfe des DisplayObject.transform.matrix3D-Objekts erzielt werden. Das DisplayObject.transform.perspectiveProjection-Objekt legt fest, wie die Anzeigeobjekte in der 3D-Perspektive gezeichnet werden. Verwenden Sie diesen Ansatz, wenn Sie 3D-Objekte animieren wollen, die hauptsächlich aus Flächen bestehen. Beispiele für diesen Ansatz sind 3D-Bildgalerien oder im 3D-Raum angeordnete 2D-Animationsobjekte.

2 Die Erzeugung von 2D-Dreiecken aus 3D-Körpern und Rendern dieser Dreiecke mit Texturen. Für diesen Ansatz müssen Sie zunächst Daten über dreidimensionale Objekte definieren und verwalten und diese Daten dann für das Rendern in zweidimensionale Dreiecke konvertieren. Diese Dreiecke werden dann Bitmaptexturen zugeordnet und anschließend mithilfe der `Graphics.drawTriangles()`-Methode auf ein Grafikobjekt gezeichnet. Praktische Beispiele für diesen Ansatz sind das Laden dreidimensionaler Modelldaten aus einer Datei und Rendern des Modells auf dem Bildschirm oder Erzeugen und Zeichnen eines 3D-Geländes in Form eines Dreieckgitters.

Erstellen und Verschieben dreidimensionaler Anzeigeobjekte

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Um ein zweidimensionales in ein dreidimensionales Anzeigeobjekt zu konvertieren, können Sie seine `z`-Eigenschaft explizit auf einen numerischen Wert setzen. Wenn Sie der `z`-Eigenschaft einen Wert zuweisen, wird ein neues Transform-Objekt für das Anzeigeobjekt erstellt. Sie können ein neues Transform-Objekt auch erstellen, indem Sie die `DisplayObject.rotationX`- oder `DisplayObject.rotationY`-Eigenschaft einstellen. Das Transform-Objekt besitzt eine `Matrix3D`-Eigenschaft, die bestimmt, wie das Anzeigeobjekt im dreidimensionalen Raum dargestellt wird.

Im folgenden Code werden die Koordinaten für ein Anzeigeobjekt mit dem Namen „leaf“ gesetzt:

```
leaf.x = 100; leaf.y = 50; leaf.z = -30;
```

Sie können diese Werte sowie davon abgeleitete Eigenschaften über die `matrix3D`-Eigenschaft des Transform-Objekts von „leaf“ anzeigen:

```
var leafMatrix:Matrix3D  = leaf.transform.matrix3D;

trace(leafMatrix.position.x);
trace(leafMatrix.position.y);
trace(leafMatrix.position.z);
trace(leafMatrix.position.length);
trace(leafMatrix.position.lengthSquared);
```

Informationen zu den Eigenschaften des Transform-Objekts finden Sie in der Beschreibung der Transform-Klasse. Informationen zu den Eigenschaften des Matrix3D-Objekts finden Sie in der Beschreibung der Matrix3D-Klasse.

Verschieben eines Objekts im 3D-Raum

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie verschieben ein Objekt in einem dreidimensionalen Raum, indem Sie die Werte der `x-, y- oder z-`Eigenschaft ändern. Wenn Sie den Wert der `z`-Eigenschaft ändern, entsteht der Eindruck, dass das Objekt dem Betrachter näher rückt oder sich vom Betrachter entfernt.

Durch den folgenden Code werden zwei Ellipsen entlang ihrer z-Achsen vor- und zurückgesetzt, indem als Reaktion auf ein Ereignis der Wert ihrer `z`-Eigenschaft geändert wird. Dabei bewegt sich `ellipse2` schneller als `ellipse1`, da ihre `z`-Eigenschaft bei jedem Bildereignis um ein Vielfaches von 20 erhöht wird, während die `z`-Eigenschaft von `ellipse1` nur um ein Vielfaches von 10 erhöht wird:

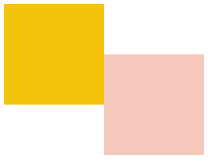
```
var depth:int = 1000;

function ellipse1FrameHandler(e:Event):void
{
    ellipse1Back = setDepth(e, ellipse1Back);
    e.currentTarget.z += ellipse1Back * 10;
}
function ellipse2FrameHandler(e:Event):void
{
    ellipse2Back = setDepth(e, ellipse1Back);
    e.currentTarget.z += ellipse1Back * 20;
}
function setDepth(e:Event, d:int):int
{
    if(e.currentTarget.z > depth)
    {
        e.currentTarget.z = depth;
        d = -1;
    }
    else if (e.currentTarget.z <  0)
    {
        e.currentTarget.z = 0;
        d = 1;
    }
}
```

Drehen eines Objekts im 3D-Raum

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können ein Objekt auf drei unterschiedliche Arten drehen, abhängig davon, wie Sie die Dreheigenschaften des Objekts einstellen: `rotationX`, `rotationY` und `rotationZ`.

Die Abbildung unten zeigt zwei Quadrate, die nicht gedreht wurden.

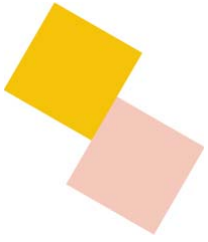


Die nächste Abbildung zeigt dieselben zwei Quadrate, wenn Sie die `rotationY`-Eigenschaft des Containers der beiden Quadrate erhöhen, sodass sie um die y-Achse gedreht werden. Durch das Drehen des Containers, oder übergeordneten Anzeigeobjekts, werden auch die beiden Quadrate gedreht:

```
container.rotationY += 10;
```

Die nächste Abbildung zeigt, was passiert, wenn Sie die `rotationX`-Eigenschaft des Containers der Quadrate setzen. Die Quadrate werden um die x-Achse gedreht.

Die nächste Abbildung zeigt, was passiert, wenn Sie die `rotationZ`-Eigenschaft des Containers der Quadrate erhöhen. Die Quadrate werden um die z-Achse gedreht.



Ein Anzeigeobjekt kann im dreidimensionalen Raum gleichzeitig verschoben und gedreht werden.

Projizieren von dreidimensionalen Objekten auf eine Fläche

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die PerspectiveProjection-Klasse im `flash.geom`-Paket bietet eine einfache Möglichkeit, beim Bewegen von Anzeigeobjekten durch den dreidimensionalen Raum eine rudimentäre Perspektive zu implementieren.

Wenn Sie für Ihren dreidimensionalen Raum nicht ausdrücklich eine perspektivische Projektion erstellen, verwendet die 3D-Engine ein Standard-PerspectiveProjection-Objekt, das im Stammelement angelegt wird und seine Eigenschaften an alle untergeordneten Elemente weitergibt.

Folgende drei Eigenschaften definieren, wie ein PerspectiveProjection-Objekt den dreidimensionalen Raum darstellt:

- `fieldOfView`
- `projectionCenter`

- `focalLength`

Durch eine Änderung des `fieldOfView`-Wertes wird automatisch auch der Wert von `focalLength` geändert und umgekehrt, da die beiden Werte voneinander abhängig sind.

Die Formel zur Berechnung des `focalLength`-Wertes bei gegebenem `fieldOfView`-Wert lautet wie folgt:

```
focalLength = stageWidth/2 * (cos(fieldOfView/2) / sin(fieldOfView/2)
```

Typischerweise würden Sie die `fieldOfView`-Eigenschaft explizit ändern.

Sichtfeld

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Mit der `fieldOfView`-Eigenschaft der PerspectiveProjection-Klasse stellen Sie das Sichtfeld ein. Dadurch können Sie erreichen, dass ein dreidimensionales Anzeigeobjekt, das sich dem Betrachter nähert, größer erscheint und ein Objekt, das sich entfernt, kleiner.

In der `fieldOfView`-Eigenschaft können Sie einen Winkel *zwischen* 0 und 180 Grad einstellen, der die Stärke der perspektivischen Projektion bestimmt. Je höher dieser Wert, desto stärker die Verzerrung eines entlang der z-Achse verschobenen Anzeigeobjekts. Bei einem niedrigen `fieldOfView`-Wert werden die Objekte nur wenig verkleinert und scheinen sich nur leicht in den Hintergrund zu bewegen. Bei einem hohen `fieldOfView`-Wert werden die Objekte stärker verzerrt und auch der Eindruck von Bewegung ist stärker. Bei einem Höchstwert von 179,9999... Grad ist das Ergebnis ein extremer Fischaugeneffekt. Der Höchstwert von `fieldOfView` beträgt 179,9999... , der Mindestwert 0,00001... Die Werte 0 und 180 sind unzulässig.

Projektionszentrum

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die `projectionCenter`-Eigenschaft repräsentiert das Projektionszentrum, d. h. den Fluchtpunkt in der perspektivischen Projektion. Der Fluchtpunkt wird als Abstand vom Standardregistrierungspunkt (0,0) definiert, der sich links oben auf der Bühne befindet.

Während sich ein Objekt weiter vom Betrachter wegzubewegen scheint, neigt es sich zum Fluchtpunkt hin und verschwindet schließlich. Stellen Sie sich eine unendlich lange Halle vor. Wenn Sie die Halle hinunterschauen, laufen die Kanten der Wände weit hinten in der Halle in einem Fluchtpunkt zusammen.

Befindet sich der Fluchtpunkt im Zentrum der Bühne, so verschwindet die Halle in einem Punkt in der Mitte. Der Standardwert für die `projectionCenter`-Eigenschaft ist die Bühnenmitte. Wenn Sie nun erreichen möchten, dass Elemente links in der Bühne erscheinen und ein dreidimensionaler Bereich rechts davon, setzen Sie `projectionCenter` auf einen Punkt rechts von der Bühne, um diesen zum Fluchtpunkt des 3D-Sichtbereichs zu machen.

Brennweite

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die `focalLength`-Eigenschaft repräsentiert die Brennweite, also den Abstand zwischen dem Standort des Betrachters (0,0,0) und der Position des Anzeigeobjekts auf seiner z-Achse.

Eine lange Brennweite ist wie ein Teleobjektiv mit Nahsicht und verkürzten Entfernungen zwischen den Objekten. Eine kurze Brennweite ist wie ein Weitwinkelobjektiv, mit dem Sie einen sehr breiten Sichtbereich bei starker Objektverzerrung erhalten. Eine mittlere Brennweite entspricht in etwa dem, was das menschliche Auge sehen kann.

Typischerweise wird die `focalLength`-Eigenschaft während der perspektivischen Transformation dynamisch neu berechnet, während sich das Anzeigeobjekt bewegt. Sie können den Wert aber auch explizit festlegen.

Standardwerte für die perspektivische Projektion

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das im Stammelement angelegte PerspectiveProjection-Standardobjekt besitzt die folgenden Werte:

- `fieldOfView: 55`
- `perspectiveCenter: stagewidth/2, stageHeight/2`
- `focalLength: stageWidth/ 2 * ( cos(fieldOfView/2) / sin(fieldOfView/2) )`

Diese Werte werden verwendet, wenn Sie nicht Ihr eigenes PerspectiveProjection-Objekt erstellen.

Sie können aber auch Ihr eigenes PerspectiveProjection-Objekt instanziieren und die Eigenschaften `projectionCenter` und `fieldOfView` selbst festlegen. In diesem Fall sind die Standardwerte des neu erstellten Objekts wie folgt, basierend auf einer Standardbühnengröße von 500 auf 500:

- `fieldOfView: 55`
- `perspectiveCenter: 250,250`
- `focalLength: 480.24554443359375`

Beispiel: Perspektivische Projektion

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Im folgenden Beispiel wird der Einsatz der perspektivischen Projektion zur Schaffung eines dreidimensionalen Raums veranschaulicht. Es wird gezeigt, wie Sie mit der `projectionCenter`-Eigenschaft den Fluchtpunkt und die perspektivische Projektion des Raums ändern können. Diese Änderung erzwingt eine Neuberechnung der Brennweite (`focalLength`) und des Sichtfeldes (`fieldOfView`) bei gleichzeitiger Verzerrung des 3D-Raums.

In diesem Beispiel geschieht Folgendes:

1. Erstellung eines Sprite-Objekts mit dem Namen `center` als Kreis mit einem Fadenkreuz
2. Zuweisung der Koordinaten des Sprite-Objekts `center` zur `projectionCenter`-Eigenschaft der `perspectiveProjection`-Eigenschaft der `transform`-Eigenschaft des Stammelements
3. Definieren von Ereignis-Listenern für mehrere Mausereignisse, die Prozeduren aufrufen, die das `projectionCenter` so ändern, dass es der Position des Objekts `center` entspricht
4. Erstellen von Rechtecken im Akkordeonstil, die die Wände des Perspektivraums bilden

Wenn Sie das Beispiel, ProjectionDragger.swf, testen, sollten Sie den Kreis auf verschiedene Positionen ziehen. Der Fluchtpunkt folgt dem Kreis und wird an der Position des Mauscursors abgesetzt, wenn Sie die Maustaste loslassen. Die Rechtecke, die den Raum umschließen, werden gestreckt und verzerrt, wenn Sie die Projektionsmitte weit von der Bühnenmitte entfernt absetzen.

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „ProjectionDragger“ finden Sie im Ordner „Samples/ProjectionDragger“.

```
package
{
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.geom.Point;
    import flash.events.*;
    public class ProjectionDragger extends Sprite
    {
        private var center : Sprite;
        private var boxPanel:Shape;
        private var inDrag:Boolean = false;

        public function ProjectionDragger():void
        {
            createBoxes();
            createCenter();
        }
        public function createCenter():void
        {
            var  centerRadius:int = 20;

            center = new Sprite();

            // circle
            center.graphics.lineStyle(1, 0x000099);
            center.graphics.beginFill(0xCCCCCC, 0.5);
            center.graphics.drawCircle(0, 0, centerRadius);
            center.graphics.endFill();
            // cross hairs
            center.graphics.moveTo(0, centerRadius);
            center.graphics.lineTo(0, -centerRadius);
            center.graphics.moveTo(centerRadius, 0);
            center.graphics.lineTo(-centerRadius, 0);
            center.x = 175;
            center.y = 175;
            center.z = 0;
            this.addChild(center);

            center.addEventListener(MouseEvent.MOUSE_DOWN, startDragProjectionCenter);
            center.addEventListener(MouseEvent.MOUSE_UP, stopDragProjectionCenter);
            center.addEventListener( MouseEvent.MOUSE_MOVE, doDragProjectionCenter);
            root.transform.perspectiveProjection.projectionCenter = new Point(center.x,
center.y);
        }
        public function createBoxes():void
        {
            // createBoxPanel();
            var boxWidth:int = 50;
            var boxHeight:int = 50;
            var numLayers:int = 12;
            var depthPerLayer:int = 50;

            // var boxVec:Vector.<Shape> = new Vector.<Shape>(numLayers);
```

```
            for (var i:int = 0; i < numLayers; i++)
            {
                this.addChild(createBox(150, 50, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xCCCCFF));
                this.addChild(createBox(50, 150, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xFFCCCC));
                this.addChild(createBox(250, 150, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xCCFFCC));
                this.addChild(createBox(150, 250, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xDDDDDD));
            }
        }

        public function createBox(xPos:int = 0, yPos:int = 0, zPos:int = 100, w:int = 50, h:int
= 50, color:int = 0xDDDDDD):Shape
        {
            var box:Shape = new Shape();
            box.graphics.lineStyle(2, 0x666666);
            box.graphics.beginFill(color, 1.0);
            box.graphics.drawRect(0, 0, w, h);
            box.graphics.endFill();
            box.x = xPos;
            box.y = yPos;
            box.z = zPos;
            return box;
        }
        public function startDragProjectionCenter(e:Event)
        {
            center.startDrag();
            inDrag = true;
        }

        public function doDragProjectionCenter(e:Event)
        {
            if (inDrag)
            {
                root.transform.perspectiveProjection.projectionCenter = new Point(center.x,
center.y);
            }
        }

        public function stopDragProjectionCenter(e:Event)
        {
            center.stopDrag();
            root.transform.perspectiveProjection.projectionCenter = new Point(center.x,
center.y);
            inDrag = false;
        }
    }
}
```

Für komplexere perspektivischen Projektionen verwenden Sie die Matrix3D-Klasse.

Durchführen komplexer 3D-Transformationen

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Matrix3D-Klasse ermöglicht Ihnen die Transformation von 3D-Punkten innerhalb eines Koordinatenraums oder die Abbildung von 3D-Punkten von einem Koordinatenraum auf einen anderen.

Sie brauchen keine Matrizenrechnung zu beherrschen, um mit der Matrix3D-Klasse zu arbeiten. Die meisten der gebräuchlichen Transformationsoperationen können mithilfe der Methoden dieser Klasse durchgeführt werden. Sie brauchen die Werte der einzelnen Elemente in der Matrix nicht explizit einzustellen oder zu berechnen.

Nachdem Sie die `z`-Eigenschaft eines Anzeigeobjekts auf einen numerischen Wert gesetzt haben, können Sie mithilfe der Matrix3D-Eigenschaft des Transform-Objekts des Anzeigeobjekts die Transformationsmatrix des Objekts abrufen:

```
var leafMatrix:Matrix3D = this.transform.matrix3D;
```

Mithilfe der Methoden des Matrix3D-Objekts können Sie eine Versetzung, Drehung, Skalierung und perspektivische Projektion an dem Anzeigeobjekt durchführen.

Verwenden Sie die Vector3D-Klasse mit ihren Eigenschaften `x`, `y` und `z` für die Verwaltung von 3D-Punkten. Die Klasse kann auch einen räumlichen Vektor in der Physik repräsentieren, der eine bestimmte Richtung und Größe besitzt. Die Methoden der Vector3D-Klasse ermöglichen die Durchführung allgemeiner Berechnungen mit räumlichen Vektoren wie Addition, Skalarprodukt oder Kreuzprodukt (auch Vektorprodukt).

Hinweis: *Die Vector3D-Klasse ist nicht mit der ActionScript-Klasse „Vector“ verwandt. Die Vector3D-Klasse umfasst Eigenschaften und Methoden für die Definition und Bearbeitung von 3D-Punkten, während die Vector-Klasse Arrays typisierter Objekte unterstützt.*

Erstellen von Matrix3D-Objekten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Für das Erstellen oder Abrufen von `Matrix3D`-Objekten gibt es drei vorrangige Verfahren:

- Verwenden der `Matrix3D()`-Konstruktormethode zur Instanziierung einer neuen Matrix. Der `Matrix3D()`-Konstruktor akzeptiert ein `Vector`-Objekt mit 16 numerischen Werten und setzt jeden dieser Werte in eine Zelle der Matrix ein. Zum Beispiel:

  ```
  var rotateMatrix:Matrix3D = new Matrix3D(1,0,0,1, 0,1,0,1, 0,0,1,1, 0,0,0,1);
  ```

- Festlegen des Wertes der `z`-Eigenschaft eines Anzeigeobjekts und anschließendes Abrufen der Transformationsmatrix aus der `transform.matrix3D`-Eigenschaft dieses Objekts.
- Abrufen des Matrix3D-Objekts, das die Anzeige von 3D-Objekten auf der Bühne steuert, indem die Methode `perspectiveProjection.toMatrix3D()` des Stammanzeigeobjekts abgerufen wird.

Anwenden mehrfacher 3D-Transformationen

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Mit einem Matrix3D-Objekt können Sie mehrere 3D-Transformationen gleichzeitig anwenden. Wenn Sie beispielsweise einen Würfel drehen, skalieren und dann verschieben wollen, könnten Sie alle drei Transformationen gesondert auf jeden Punkt des Würfels anwenden. Allerdings ist es sehr viel effizienter, mehrere Transformationen in einem Matrix3D-Objekt vorauszuberechnen und dann eine Matrixtransformation auf jeden der Punkte anzuwenden.

Hinweis: *Dabei ist die Reihenfolge wichtig, in der die Matrixtransformationen angewendet werden. Matrixberechnungen sind nicht kommutativ. Wenn beispielsweise erst eine Drehung und dann eine Versetzung angewendet wird, entsteht ein anderes Ergebnis als wenn Sie erst dieselbe Versetzung und danach dieselbe Drehung anwenden.*

Im folgenden Beispiel werden zwei Wege aufgezeigt, um mehrere 3D-Transformationen durchzuführen.

```
package {
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.display.Graphics;
    import flash.geom.*;

public class Matrix3DTransformsExample extends Sprite
    {
        private var rect1:Shape;
        private var rect2:Shape;

public function Matrix3DTransformsExample():void
        {
            var pp:PerspectiveProjection = this.transform.perspectiveProjection;
            pp.projectionCenter = new Point(275,200);
            this.transform.perspectiveProjection = pp;

            rect1 = new Shape();
            rect1.x = -70;
            rect1.y = -40;
            rect1.z = 0;
            rect1.graphics.beginFill(0xFF8800);
            rect1.graphics.drawRect(0,0,50,80);
            rect1.graphics.endFill();
            addChild(rect1);

            rect2 = new Shape();
            rect2.x = 20;
            rect2.y = -40;
            rect2.z = 0;
            rect2.graphics.beginFill(0xFF0088);
            rect2.graphics.drawRect(0,0,50,80);
            rect2.graphics.endFill();
            addChild(rect2);
```

```
            doTransforms();
        }

        private function doTransforms():void
        {
            rect1.rotationX = 15;
            rect1.scaleX = 1.2;
            rect1.x += 100;
            rect1.y += 50;
            rect1.rotationZ = 10;

            var matrix:Matrix3D = rect2.transform.matrix3D;
            matrix.appendRotation(15, Vector3D.X_AXIS);
            matrix.appendScale(1.2, 1, 1);
            matrix.appendTranslation(100, 50, 0);
            matrix.appendRotation(10, Vector3D.Z_AXIS);
            rect2.transform.matrix3D = matrix;
        }
    }
}
```

In der `doTransforms()`-Methode verwendet der erste Codeblock die DisplayObject-Eigenschaften zum Ändern der Drehung, Skalierung und Position eines Rechtecks. Der zweite Codeblock verwendet die Methoden der Matrix3D-Klasse, um dieselben Transformationen durchzuführen.

Der wichtigste Vorteil beim Verwenden der `Matrix3D`-Methoden liegt darin, dass alle Berechnungen zuerst in der Matrix stattfinden. Anschließend werden sie nur einmal auf das Anzeigeobjekt angewendet, wenn seine `transform.matrix3D`-Eigenschaft gesetzt ist. Die Einstellung der DisplayObject-Eigenschaften macht das Lesen des Quellcodes etwas einfacher. Allerdings sind jedes Mal, wenn eine Dreh- oder Skalierungseigenschaft gesetzt wird, mehrere Berechnungen und Änderungen an mehreren Anzeigeobjekteigenschaften erforderlich.

Wenn Ihr Code dieselben komplexen Transformationen an Anzeigeobjekten mehrmals vornimmt, sollten Sie das Matrix3D-Objekt als Variable speichern und dann immer wieder anwenden.

Verwenden von 3DMatrix-Objekten zur Neuanordnung der Anzeige

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wie bereits weiter vorne erwähnt, bestimmt die Reihenfolge der Ebenen in der Anzeigeliste die Anzeigereihenfolge, ungeachtet ihrer relativen z-Achsen. Wenn Ihre Animation die Eigenschaften von Anzeigeobjekten in einer Reihenfolge transformiert, die von der Reihenfolge in der Anzeigeliste abweicht, kann es passieren, dass der Betrachter eine Anordnung gemäß der Anzeigeliste sieht, die der z-Achsenanordnung widerspricht. Das bedeutet, dass ein Objekt, das weiter entfernt vom Betrachter angezeigt werden sollte, vor einem Objekt erscheint, das näher beim Betrachter liegt.

Um sicherzustellen, dass die Anordnung von dreidimensionalen Anzeigeobjekten der relativen Tiefe der Objekte entspricht, sollten Sie einen Ansatz wie den folgenden beachten:

1. Verwenden Sie die `getRelativeMatrix3D()`-Methode des Transform-Objekts, um die relativen `z-Achsen` des untergeordneten 3D-Anzeigeobjekts abzurufen.
2. Entfernen Sie die Objekte mithilfe der `removeChild()`-Methode aus der Anzeigeliste.
3. Sortieren Sie die Anzeigeobjekte aufgrund ihrer relativen z-Achsenwerte.
4. Fügen Sie die untergeordneten Objekte mithilfe der `addChild()`-Methode in umgekehrter Reihenfolge wieder in die Anzeigeliste ein.

Diese Neuanordnung stellt sicher, dass Ihre Objekte in Übereinstimmung mit den relativen z-Achsen angezeigt werden.

Im folgenden Code wird die korrekte Anzeige der sechs Flächen eines Quaders erzwungen. Die Flächen des Quaders werden neu angeordnet, nachdem Drehungen darauf angewendet worden sind.

```
public var faces:Array; . . .

public function ReorderChildren()
{
    for(var ind:uint = 0; ind < 6; ind++)
    {
        faces[ind].z = faces[ind].child.transform.getRelativeMatrix3D(root).position.z;
        this.removeChild(faces[ind].child);
    }
    faces.sortOn("z", Array.NUMERIC | Array.DESCENDING);
    for (ind = 0; ind < 6; ind++)
    {
        this.addChild(faces[ind].child);
    }
}
```

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Anwendungsdateien befinden sich im Ordner „Samples/ReorderByZ “.

Verwenden von Dreiecken für 3D-Effekte

Flash Player 10 und höher, Adobe AIR 1.5 und höher

In ActionScript führen Sie Bitmaptransformationen mithilfe der `Graphics.drawTriangles()`-Methode durch, da 3D-Modelle durch eine Sammlung von Dreiecken im Raum dargestellt werden. (Allerdings unterstützen Flash Player und AIR keinen Tiefenpuffer, sodass Anzeigeobjekte im Prinzip weiterhin flach, d. h. zweidimensional, sind. Dies wird beschrieben unter „Informationen zu 3D-Anzeigeobjekten in Flash Player und der AIR-Laufzeitumgebung“ auf Seite 376.) Die `Graphics.drawTriangles()`-Methode funktioniert ähnlich wie die `Graphics.drawPath()`-Methode, da zum Zeichnen eines Dreieckpfads ebenfalls ein Satz an Koordinaten benötigt wird.

Um sich mit der Arbeit mit der `Graphics.drawPath()`-Methode vertraut zu machen, sollten Sie den Abschnitt „Zeichenpfade“ auf Seite 251 lesen.

Die `Graphics.drawTriangles()`-Methode verwendet die Syntax „Vector.<Number>“, um die Position der Punkte für den Dreieckpfad anzugeben:

```
drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null, uvtData:Vector.<Number>
= null, culling:String = "none"):void
```

Der erste Parameter von `drawTriangles()` ist der einzige obligatorische Parameter, nämlich der Parameter `vertices`. Dieser Parameter ist ein Zahlenvektor, der die Koordinaten definiert, aus denen die Dreiecke gezeichnet werden. Alle drei Koordinatensets (sechs Zahlen) repräsentieren einen Dreieckpfad. Ohne den `indices`-Parameter sollte die Länge des Vektors immer das Sechsfache betragen, da jedes Dreieck drei Koordinatenpaare benötigt (d. h. drei Sätze an x/y-Werten). Zum Beispiel:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([
        10,10,  100,10,  10,100,
        110,10, 110,100, 20,100]));
```

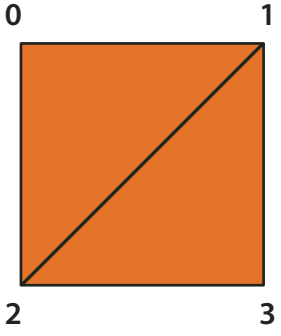
Keine dieser Dreiecke teilen sich einen Punkt, aber wenn das der Fall wäre, könnte der Wert mithilfe des zweiten `drawTriangles()`-Parameters, `indices`, verwendet werden, um die Werte im Vektor `vertices` für mehrere Dreiecke wiederzuverwenden.

Wenn Sie den `indices`-Parameter verwenden, sollten Sie sich darüber im Klaren sein, dass die `indices`-Werte Punktindizes sind und keine Indizes, die sich direkt auf die `vertices`-Arrayelemente beziehen. Anders ausgedrückt: Eine durch `indices` im `vertices`-Vektor definierte Indexposition ist in Wahrheit die echte Indexposition dividiert durch 2. Für den dritten Punkt eines `vertices`-Vektors verwenden Sie beispielsweise den `indices`-Wert 2, obwohl der erste numerische Wert dieses Punkts bei der Vektorindexposition 4 beginnt.

Sie können zum Beispiel zwei Dreiecke mithilfe des `indices`-Parameters so zusammenführen, dass sie sich in der Diagonale eines Vierecks eine Kante teilen:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([10,10, 100,10, 10,100, 100,100]),
    Vector.<int>([0,1,2, 1,3,2]));
```

Beachten Sie, dass im `vertices`-Vektor nur vier Punkte angegeben wurden, obwohl ein Quadrat unter Verwendung von zwei Dreiecken gezeichnet wurde. Durch die Verwendung von `indices` werden die beiden Punkte, die beide Dreiecke gemeinsam haben, für jedes Dreieck verwendet. Dadurch reduziert sich die Gesamtzahl der Vertizes von 6 (12 Zahlen) auf 4 (8 Zahlen):



Ein mithilfe des vertices-Parameters aus zwei Dreiecken gezeichnetes Quadrat

Je größer die Dreieckgitter, desto nützlicher wird diese Technik, da dabei die meisten Punkte von mehreren Dreiecken geteilt werden.

Auf Dreiecke können alle Füllungen angewendet werden. Die Füllungen werden auf das resultierende Dreieckgitter wie auf jede sonstige Form angewendet.

Transformieren von Bitmaps

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Bitmaptransformationen schaffen die Illusion von Perspektive oder „Textur“ auf einem dreidimensionalen Objekt. Insbesondere können Sie eine Bitmap zum Fluchtpunkt hin verzerren, sodass das Bild kleiner zu werden scheint, während es sich dem Fluchtpunkt nähert. Sie können auch eine zweidimensionale Bitmap verwenden, um eine Oberfläche für ein dreidimensionales Objekt zu erstellen und so den Eindruck erwecken als ob die Textur das dreidimensionale Objekt umwickelte.

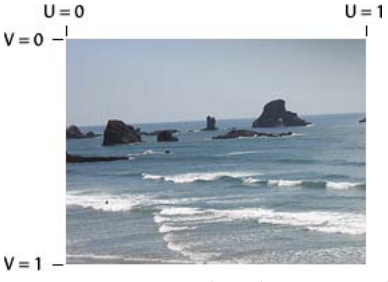
Eine zweidimensionale Oberfläche, die einen Fluchtpunkt verwendet und ein von einer Bitmap umwickeltes dreidimensionales Objekt

UV-Zuordnung

Flash Player 10 und höher, Adobe AIR 1.5 und höher

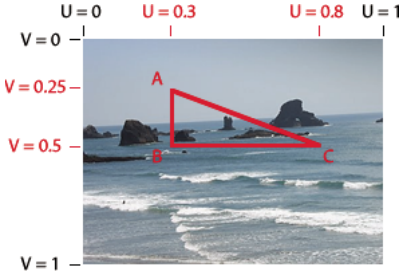
Wenn Sie erst einmal mit der Arbeit mit Texturen begonnen haben, werden Sie auch den uvtData-Parameter von `drawTriangles()` verwenden wollen. Dieser Parameter ermöglicht es Ihnen, die UV-Zuordnung für Bitmapfüllungen einzurichten.

Die UV-Zuordnung ist ein Verfahren zur Texturierung von Objekten. Sie ist von zwei Werten abhängig, einem horizontalen U-Wert (x) und einem vertikalen V-Wert (y). Diese basieren nicht auf Pixelwerten, sondern auf Prozentwerten. 0 U und 0 V bezeichnet die Bildecke links oben und 1 U und 1 V die Bildecke rechts unten:



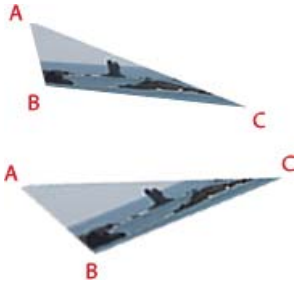
Die Positionen UV 0 und 1 auf einem Bitmapbild

Die Vektoren eines Dreiecks können als UV-Koordinaten angegeben werden, um sich mit den entsprechenden Positionen in einem Bild zu verbinden:



Die UV-Koordinaten eines Dreieckbereichs aus einem Bitmapbild

Die UV-Werte bleiben mit den Punkten des Dreiecks konsistent:



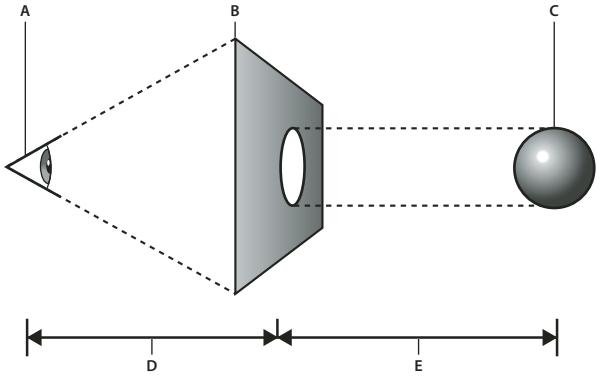
Die Vertizes des Dreiecks werden verschoben und die Bitmap wird verzerrt, um den UV-Wert für einen einzelnen Punkt unverändert zu lassen:

Während auf das mit der Bitmap verbundene Dreieck ActionScript-3D-Transformationen angewendet werden, wird das Bitmapbild aufgrund der UV-Werte auf das Dreieck angewendet. Anstatt also Matrixberechnungen zu verwenden, werden einfach die UV-Werte gesetzt oder angepasst, um einen dreidimensionalen Effekt zu erschaffen.

Die `Graphics.drawTriangles()`-Methode akzeptiert auch eine optionale Information für dreidimensionale Transformationen: den T-Wert. Der T-Wert in „uvtData“ repräsentiert die 3D-Perspektive, oder genauer: den Skalierungsfaktor des verbundenen Vertex. Die UVT-Zuordnung ergänzt die UV-Zuordnung um perspektivische Korrekturen. Wenn ein Objekt beispielsweise in einem 3D-Raum vom Betrachterstandpunkt entfernt positioniert wird, sodass es nur 50 % der „Originalgröße“ zu haben scheint, würde der T-Wert dieses Objekts 0,5 betragen. Da Dreiecke gezeichnet werden, um die Objekte im 3D-Raum zu repräsentieren, bestimmen deren Positionen auf der z-Achse deren T-Werte. Die Gleichung, die den T-Wert bestimmt, lautet wie folgt:

```
T = focalLength/(focalLength + z);
```

In dieser Gleichung steht „focalLength“ für die Brennweite oder eine berechnete Bildschirmposition, die bestimmt, wie stark die Perspektive in dieser Sicht ist.



Die Brennweite und der z-Wert
***A.** Standpunkt **B.** Bildschirm **C.** 3D-Objekt **D.** focalLength-Wert (Brennweite) **E.** z-Wert*

Der Wert von T wird verwendet, um die Grundformen zu verkleinern, sodass sie scheinbar weiter entfernt sind. Mit diesem Wert werden in der Regel 3D-Punkte auf 2D-Punkte abgebildet. Im Fall der UVT-Daten wird er auch dazu verwendet, um eine Bitmap zwischen den Punkten innerhalb eines Dreiecks mit Perspektive zu skalieren.

Wenn Sie UVT-Werte definieren, folgt der T-Wert direkt den UV-Werten für einen Vertex. Wird der T-Wert miteinbezogen, sind alle drei Werte im `uvtData`-Parameter (U, V und T) stimmig mit den beiden Werten im `vertices`-Parameter (x und y). Mit UV-Werten allein ist uvtData.length == vertices.length. Unter Miteinbeziehung eines T-Werts ist uvtData.length == 1,5*vertices.length.

Im folgenden Beispiel wird gezeigt, wie eine Fläche mithilfe von TVT-Daten im 3D-Raum gedreht werden kann. In diesem Beispiel wird das Bild „ocean.jpg“ mithilfe der „Hilfsklasse“ „ImageLoader“ geladen, sodass es dem BitmapData-Objekt zugewiesen werden kann.

Hier der Quellcode der ImageLoader-Klasse (speichern Sie diesen Code in eine Datei mit dem Namen „ImageLoader.as“):

```
package {
    import flash.display.*
    import flash.events.*;
    import flash.net.URLRequest;
    public class ImageLoader extends Sprite {
        public var url:String;
        public var bitmap:Bitmap;
    public function ImageLoader(loc:String = null) {
        if (loc != null){
            url = loc;
            loadImage();
        }
    }
    public function loadImage():void{
        if (url != null){
            var loader:Loader = new Loader();
            loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onComplete);
            loader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, onIoError);

                var req:URLRequest = new URLRequest(url);
                loader.load(req);
            }
        }

    private function onComplete(event:Event):void {
            var loader:Loader = Loader(event.target.loader);
            var info:LoaderInfo = LoaderInfo(loader.contentLoaderInfo);
            this.bitmap = info.content as Bitmap;
            this.dispatchEvent(new Event(Event.COMPLETE));
    }

    private function onIoError(event:IOErrorEvent):void {
            trace("onIoError: " + event);
    }
    }
}
```

Es folgt das ActionScript-Codebeispiel, das Dreiecke, die UV-Zuordnung und T-Werte verwendet, um den Eindruck zu vermitteln, dass das Bild zum Fluchtpunkt hin kleiner wird und sich dreht. Speichern Sie diesen Code in eine Datei mit dem Namen „Spinning3dOcean.as“:

```
package {
    import flash.display.*
    import flash.events.*;
    import flash.utils.getTimer;

    public class Spinning3dOcean extends Sprite {
        // plane vertex coordinates (and t values)
        var x1:Number = -100,y1:Number = -100,z1:Number = 0,t1:Number = 0;
        var x2:Number = 100,y2:Number = -100,z2:Number = 0,t2:Number = 0;
        var x3:Number = 100,y3:Number = 100,z3:Number = 0,t3:Number = 0;
        var x4:Number = -100,y4:Number = 100,z4:Number = 0,t4:Number = 0;
        var focalLength:Number = 200;
        // 2 triangles for 1 plane, indices will always be the same
        var indices:Vector.<int>;

        var container:Sprite;

        var bitmapData:BitmapData; // texture
        var imageLoader:ImageLoader;
        public function Spinning3dOcean():void {
            indices =  new Vector.<int>();
            indices.push(0,1,3, 1,2,3);

            container = new Sprite(); // container to draw triangles in
            container.x = 200;
            container.y = 200;
            addChild(container);

            imageLoader = new ImageLoader("ocean.jpg");
            imageLoader.addEventListener(Event.COMPLETE, onImageLoaded);
        }
        function onImageLoaded(event:Event):void {
            bitmapData = imageLoader.bitmap.bitmapData;
            // animate every frame
            addEventListener(Event.ENTER_FRAME, rotatePlane);
        }
        function rotatePlane(event:Event):void {
            // rotate vertices over time
            var ticker = getTimer()/400;
            z2 = z3 = -(z1 = z4 = 100*Math.sin(ticker));
            x2 = x3 = -(x1 = x4 = 100*Math.cos(ticker));

            // calculate t values
```

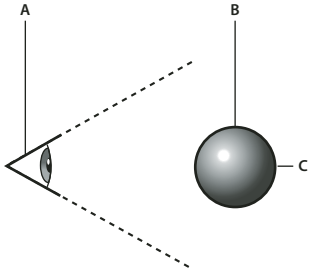
```
            t1 = focalLength/(focalLength + z1);
            t2 = focalLength/(focalLength + z2);
            t3 = focalLength/(focalLength + z3);
            t4 = focalLength/(focalLength + z4);

            // determine triangle vertices based on t values
            var vertices:Vector.<Number> = new Vector.<Number>();
            vertices.push(x1*t1,y1*t1, x2*t2,y2*t2, x3*t3,y3*t3, x4*t4,y4*t4);
            // set T values allowing perspective to change
            // as each vertex moves around in z space
            var uvtData:Vector.<Number> = new Vector.<Number>();
            uvtData.push(0,0,t1, 1,0,t2, 1,1,t3, 0,1,t4);

            // draw
            container.graphics.clear();
            container.graphics.beginBitmapFill(bitmapData);
            container.graphics.drawTriangles(vertices, indices, uvtData);
        }
    }
}
```

Um dieses Beispiel zu testen, speichern Sie diese beiden Klassendateien im selben Verzeichnis als ein Bild mit dem Namen „ocean.jpg“. Sie können sehen, wie die Orignalbitmap so transformiert wird, dass sie den Eindruck vermittelt, als ob sie im 3D-Raum in der Ferne verschwindet und sich dreht.

Culling

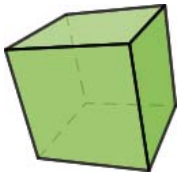
Flash Player 10 und höher, Adobe AIR 1.5 und höher

Culling ist ein Verfahren zur Bestimmung der Oberflächen eines dreidimensionalen Objekts, die nicht gerendert werden sollen, da sie vom aktuellen Betrachterstandpunkt aus nicht zu sehen sind. Im dreidimensionalen Raum ist die Oberfläche der Rückseite eines dreidimensionalen Objekts vom Sichtbereich ausgeschlossen.



Die Rückseite eines 3D-Objekts ist vom Sichtbereich ausgeschlossen
***A.** Standpunkt **B.** 3D-Objekt **C.** Rückseite des 3D-Objekts*

Normalerweise werden immer alle Dreiecke gerendert, unabhängig von ihrer Größe, Form oder Position. Das Cullingverfahren stellt sicher, dass Flash Player oder AIR Ihre 3D-Objekte korrekt rendert. Für kürzere Renderzeiten sollten bestimmte Dreiecke vom Renderprozess ausgeschlossen werden. Stellen Sie sich z. B. einen Würfel vor, der sich im Raum dreht. Egal, zu welchem Zeitpunkt, Sie werden nie mehr als drei Seiten dieses Würfels zu Gesicht bekommen, da sich die anderen Seiten auf der Rückseite des Würfels befinden. Und da diese Seiten verdeckt sind, sollte der Renderer sie auch nicht berechnen und zeichnen. Ohne Culling rendert Flash Player oder AIR sowohl die Vorder- als auch die Rückseite des Würfels.



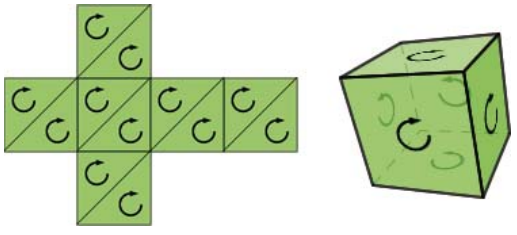
Ein Würfel hat Seiten, die vom aktuellen Betrachterstandpunkt nicht zu sehen sind

Daher besitzt die `Graphics.drawTriangles()`-Methode einen vierten Parameter zum Festlegen des Cullingwertes:

```
public function drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null,
uvtData:Vector.<Number> = null, culling:String = "none"):void
```

Dieser Parameter gehört zur `TriangleCulling`-Aufzählungsklasse und kann die Werte `TriangleCulling.NONE`, `TriangleCulling.POSITIVE` und `TriangleCulling.NEGATIVE` annehmen. Diese Werte sind abhängig von der Richtung des Dreieckpfads, der die Oberfläche des Objekts definiert. Die ActionScript-API, die das Culling steuert, setzt voraus, dass alle nach außen gerichteten Dreiecke einer dreidimensionalen Form in derselben Pfadrichtung gezeichnet werden. Wenn ein Dreieck umgedreht wird, ändert sich auch dessen Pfadrichtung. An diesem Punkt kann das Dreieck „ausgelesen“ und vom Renderprozess ausgeschlossen werden.

Der `TriangleCulling`-Wert `POSITIVE` schließt alle Dreiecke mit positiver Pfadrichtung (im Uhrzeigersinn) aus. Der `TriangleCulling`-Wert `NEGATIVE` schließt alle Dreiecke mit negativer Pfadrichtung (entgegen dem Uhrzeigersinn) aus. Bei einem Würfel haben die Oberflächen der Vorderseite eine positive Pfadrichtung und die der Rückseite eine negative:



Ein „aufgeklappter“ Würfel, in dem die Pfadrichtung angezeigt wird. Im „zusammengeklappten“ Zustand ist die Pfadrichtung der Rückseite umgekehrt

Um zu sehen, wie das Cullingverfahren funktioniert, beginnen Sie mit dem Beispiel aus dem Abschnitt „UV-Zuordnung“ auf Seite 390 und setzen Sie den culling-Parameter der `drawTriangles()`-Methode auf `TriangleCulling.NEGATIVE`:

```
container.graphics.drawTriangles(vertices, indices, uvtData, TriangleCulling.NEGATIVE);
```

Beachten Sie, dass die jeweilige Rückseite des Bildes beim Drehen des Objekts nicht dargestellt wird.

Kapitel 20: Grundlagen der Textverarbeitung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Verwenden Sie zum Anzeigen von Text am Bildschirm in Adobe® Flash® Player oder Adobe® AIR™ eine Instanz der TextField-Klasse oder die Flash Text Engine-Klassen. Diese Klassen ermöglichen Ihnen, Text zu erstellen, anzuzeigen und zu formatieren. Alternativ können Sie das Text Layout Framework (TLF) verwenden, eine Komponentenbibliothek, die auf der Flash Text Engine-Klasse basiert, aber für einfache Bedienung konzipiert wurde. Auf mobilen Geräten können Sie die StageText-Klasse für die Texteingabe verwenden.

Sie können spezifische Inhalte für Textfelder festlegen oder die Quelle für den Text angeben und dann die Darstellung dieses Textes einstellen. Sie können zudem auf Benutzerereignisse reagieren, z. B. wenn Benutzer Text eingeben oder auf einen Hyperlink klicken.

Sowohl die TextField-Klasse als auch die Flash Text Engine-Klassen ermöglichen es Ihnen, Text in Flash Player und AIR anzuzeigen und zu verwalten. Mithilfe der TextField-Klasse können Sie Textobjekte zur Anzeige und Eingabe erstellen. Die TextField-Klasse bietet die Grundlage für andere textbasierte Komponenten, zum Beispiel TextArea und TextInput. Sie könnend die TextFormat-Klasse verwenden, um die Zeichen- und Absatzformatierung für TextField-Objekte festzulegen, und Sie können mit der Textfield.styleSheet-Eigenschaft und der StyleSheet-Klasse Cascading Style Sheets (CSS) anwenden. Sie können HTML-formatierten Text, der eingebettete Medien (Film-Clips, SWF-Dateien, GIF-Dateien, PNG-Dateien und JPEG-Dateien) enthalten kann, direkt einem Textfeld zuweisen.

Die ab Flash Player 10 und Adobe AIR 1.5 verfügbare Flash Text Engine bietet eine elementare Unterstützung für anspruchsvolle Steuerungsaufgaben im Hinblick auf Textmetrik, Textformatierung und bidirektionalen Text. Sie bietet zudem einen verbesserten Textfluss und eine erweiterte Sprachunterstützung. Sie können die Flash Text Engine zwar zur Erstellung und Verwaltung von Textelementen verwenden, sie wurde jedoch in erster Linie als Grundlage für die Erstellung von Textverarbeitungskomponenten entworfen und erfordert ein höheres Maß an Programmierkenntnissen. Das Text Layout Framework umfasst eine Textverarbeitungskomponente auf Grundlage der Flash Text Engine und bietet eine einfachere Möglichkeit, die erweiterten Funktionen der neuen Text Engine zu verwenden. Das Text Layout Framework ist eine erweiterbare Bibliothek, die komplett in ActionScript 3.0 geschrieben wurde. Sie können die vorhandene TLF-Komponente verwenden oder mit dem Framework Ihre eigene Textkomponente erstellen.

Die StageText-Klasse, die ab AIR 3 verfügbar ist, stellt ein natives Texteingabefeld bereit. Da dieses Feld vom Betriebssystem des Geräts bereitgestellt wird, sind die Benutzer des Geräts damit am besten vertraut. Eine StageText-Instanz ist kein Anzeigeobjekt. Anstatt sie der Anzeigeliste hinzuzufügen, weisen Sie einer Instanz eine Bühne und einen Anzeigebereich auf dieser Bühne zu, der Viewport genannt wird. Die StageText-Instanz wird vor allen Anzeigeobjekten angezeigt.

Weitere Informationen zu diesen Themen finden Sie hier:

- „Verwenden der TextField-Klasse“ auf Seite 399
- „Verwenden der Flash Text Engine“ auf Seite 425
- „Verwenden des Text Layout Framework“ auf Seite 455
- Native text input with StageText

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe aufgeführt, die mit der Verarbeitung von Text zu tun haben:

Cascading Style Sheets Eine Standardsyntax zur Angabe von Stilen und Formatierungen für Inhalte im XML-Format (oder HTML-Format).

Geräteschriftart Eine Schriftart, die auf dem Computer des Benutzers installiert ist.

Dynamisches Textfeld Ein Textfeld, dessen Inhalt von ActionScript, jedoch nicht durch Benutzereingaben geändert werden kann.

Eingebettete Schriftart Eine Schriftart, deren Zeichenkonturdaten in der SWF-Datei der Anwendung gespeichert sind.

HTML-Text Der mit ActionScript in ein Textfeld eingegebene Text, der neben dem eigentlichen Textinhalt auch HTML-Formatierungstags enthält.

Eingabetextfeld Ein Textfeld, dessen Inhalt entweder durch Benutzereingaben oder von ActionScript geändert werden kann.

Kerning Ein Verfahren, mit dem der Abstand zwischen bestimmten Zeichenpaaren vergrößert bzw. verkleinert wird, um den Text lesbarer zu gestalten.

Statisches Textfeld Ein mit dem Authoring-Tool erstelltes Textfeld, dessen Inhalt nicht geändert werden kann, wenn die SWF-Datei ausgeführt wird.

Textzeilenmetrik Maße verschiedener Bereiche des Textinhalts in einem Textfeld, z. B. Grundlinie des Textes, Höhe der Zeichen, Größe der Unterlängen (Teil der Kleinbuchstaben unterhalb der Grundlinie) usw.

Zeichenabstand Ein Verfahren, mit dem der Abstand zwischen Buchstabengruppen oder Textblöcken vergrößert bzw. verkleinert wird, um die Dichte zu verringern und den Text lesbarer zu gestalten.

Kapitel 21: Verwenden der TextField-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können eine Instanz der TextField-Klasse verwenden, um in Adobe® Flash® Player oder Adobe® AIR™ Text anzuzeigen oder auf dem Bildschirm ein Texteingabefeld zu erstellen. Die TextField-Klasse ist die Grundlage für andere textbasierte Komponenten, zum Beispiel TextArea- oder TextInput-Komponenten.

Inhalte für Textfelder können vorab in der SWF-Datei angegeben, aus einer Textdatei oder Datenbank geladen oder interaktiv von Benutzern in der Anwendung eingegeben werden. Innerhalb eines Textfelds kann der Text als formatierter HTML-Inhalt mit eingebetteten Grafiken dargestellt werden. Nachdem Sie eine Instanz eines Textfeldes erstellt haben, können Sie mit flash.text-Klassen, z. B. TextFormat und StyleSheet, das Erscheinungsbild des Textes steuern. Das flash.text-Paket enthält fast alle Klassen für das Erstellen, Verwalten und Formatieren von Text in ActionScript.

Sie können Text formatieren, indem Sie die Formatierung mit einem TextFormat-Objekt definieren und dieses Objekt dem entsprechenden Textfeld zuweisen. Wenn das Textfeld HTML-Text enthält, können Sie ein StyleSheet-Objekt auf das Textfeld anwenden, um bestimmten Teilen des Textfeldinhalts Stile zuzuweisen. Das TextFormat-Objekt und das StyleSheet-Objekt enthalten Eigenschaften, mit denen das Erscheinungsbild des Textes definiert wird, beispielsweise die Farbe, Größe und Schriftstärke. Mit dem TextFormat-Objekt werden die Eigenschaften dem gesamten Inhalt eines Textfelds oder einem bestimmten Textbereich zugewiesen. Beispielsweise kann innerhalb desselben Textfelds ein Satz fett und rot formatiert werden und der nächste Satz kursiv und blau.

Neben den Klassen im flash.text-Paket können Sie mithilfe der flash.events.TextEvent-Klasse auf textbezogene Benutzeraktionen reagieren.

Verwandte Hilfethemen

„Zuweisen von Textformaten“ auf Seite 407

„Anzeigen von HTML-Text“ auf Seite 401

„Anwenden von Cascading Style Sheets“ auf Seite 408

Anzeigen von Text

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Obwohl in Authoring-Tools wie Adobe Flash Builder oder Flash Professional mehrere Optionen zum Anzeigen von Text verfügbar sind, einschließlich textbezogener Komponenten oder Textwerkzeuge, erfolgt die Textanzeige im Programmcode hauptsächlich über Textfelder.

Texttypen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der Typ des Textes innerhalb eines Textfelds ist durch die jeweilige Quelle charakterisiert.

- Dynamischer Text

  Dynamischer Text ist Inhalt, der aus einer externen Quelle geladen wird, z. B. aus einer Textdatei, einer XML-Datei oder einem Remote-Webservice.

- Eingabetext

  Eingabetext ist jeder von einem Benutzer eingegebene Text oder dynamischer Text, der von Benutzern bearbeitet werden kann. Sie können ein Stylesheet zum Formatieren von Eingabetext einrichten oder die flash.text.TextFormat-Klasse verwenden, um dem Eingabetext im Textfeld Eigenschaften zuzuweisen. Weitere Informationen finden Sie unter „Erfassen von Texteingaben“ auf Seite 405.

- Statischer Text

  Statischer Text wird nur von Flash Professional erstellt. Mit ActionScript 3.0 können Sie keine statische Textinstanz erstellen. Sie können jedoch ActionScript-Klassen, wie StaticText und TextSnapshot, verwenden, um eine vorhandene statische Textinstanz zu bearbeiten. Weitere Informationen finden Sie unter „Verwenden von statischem Text“ auf Seite 414.

Ändern der Inhalte von Textfeldern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können dynamischen Text definieren, indem Sie der `flash.text.TextField.text`-Eigenschaft einen String zuweisen. Das direkte Zuweisen eines Strings zu der Eigenschaft geschieht wie folgt:

```
myTextField.text = "Hello World";
```

Sie können auch wie im folgenden Beispiel der `text`-Eigenschaft den Wert einer im Skript definierten Variablen zuweisen:

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class TextWithImage extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myText:String = "Hello World";

        public function TextWithImage()
        {
            addChild(myTextBox);
            myTextBox.text = myText;
        }
    }
}
```

Alternativ können Sie der `text`-Eigenschaft den Wert einer Remote-Variablen zuweisen. Zum Laden von Textwerten aus Remote-Quellen stehen drei Optionen zur Auswahl:

- Mit den Klassen flash.net.URLLoader und flash.net.URLRequest werden Variablen für den Text von einem lokalen oder entfernten Speicherort geladen.
- Das `FlashVars`-Attribut ist in die HTML-Seite mit der SWF-Datei eingebettet und kann Werte für Textvariablen enthalten.
- Mit der flash.net.SharedObject-Klasse wird die dauerhafte Speicherung von Werten verwaltet. Weitere Informationen finden Sie unter „Speichern lokaler Daten“ auf Seite 747.

Anzeigen von HTML-Text

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die flash.text.TextField-Klasse verfügt über eine `htmlText`-Eigenschaft, mit der Sie angeben können, dass der Textstring HTML-Tags zum Formatieren des Inhalts enthält. Wie im folgenden Beispiel zu sehen ist, müssen Sie den Stringwert zur `htmlText`-Eigenschaft (und nicht zur `text`-Eigenschaft) zuweisen, damit der Text in Flash Player oder AIR als HTML dargestellt wird:

```
var myText:String = "<p>This is <b>some</b> content to <i>render</i> as <u>HTML</u> text.</p>";
myTextBox.htmlText = myText;
```

Flash Player und AIR unterstützen einen Teil der HTML-Tags und HTML-Entitäten für die `htmlText`-Eigenschaft. Die Beschreibung der `flash.text.TextField.htmlText`-Eigenschaft im ActionScript 3.0-Referenzhandbuch enthält ausführliche Informationen zu den unterstützten HTML-Tags und HTML-Entitäten.

Nachdem Sie den Inhalt mit der `htmlText`-Eigenschaft angegeben haben, können Sie die entsprechende Formatierung mithilfe von Stylesheets oder des `textformat`-Tags festlegen. Weitere Informationen finden Sie unter „Formatieren von Text“ auf Seite 407.

Verwenden von Bildern in Textfeldern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein weiterer Vorteil der Anzeige von Inhalten als HTML-Text besteht darin, dass Sie Bilder in das Textfeld einfügen können. Mithilfe des `img`-Tags können Sie auf lokale oder entfernte Bilder verweisen und diese innerhalb des entsprechenden Textfelds anzeigen.

Im folgenden Beispiel wird ein Textfeld mit dem Namen `myTextBox` erstellt und in den Anzeigetext wird das JPG-Bild eines Auges eingefügt, das sich im selben Verzeichnis wie die SWF-Datei befindet:

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class TextWithImage extends Sprite
    {
        private var myTextBox:TextField;
        private var myText:String = "<p>This is <b>some</b> content to <i>test</i> and
<i>see</i></p><p><img src='eye.jpg' width='20' height='20'></p><p>what can be
rendered.</p><p>You should see an eye image and some <u>HTML</u> text.</p>";

        public function TextWithImage()
        {
            myTextBox.width = 200;
            myTextBox.height = 200;
            myTextBox.multiline = true;
            myTextBox.wordWrap = true;
            myTextBox.border = true;

            addChild(myTextBox);
            myTextBox.htmlText = myText;
        }
    }
}
```

Das `img`-Tag unterstützt JPEG-, GIF-, PNG- und SWF-Dateien.

Bildlauf in Textfeldern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In vielen Fällen ist der Text länger als das Textfeld, in dem er angezeigt wird. Möglicherweise verwenden Sie auch ein Eingabefeld, in dem Benutzer mehr Text eingeben können, als auf einmal angezeigt werden kann. Zum Verwalten umfangreicher Inhalte können Sie die mit dem vertikalen und horizontalen Bildlauf verknüpften Eigenschaften der flash.text.TextField-Klasse verwenden.

Dazu gehören `TextField.scrollV`, `TextField.scrollH`, `maxScrollV` und `maxScrollH`. Verwenden Sie diese Eigenschaften, um auf Ereignisse wie Mausklicks oder einen Tastendruck zu reagieren.

Im folgenden Beispiel wird ein Textfeld fester Größe erstellt, das mehr Text enthält, als auf einmal angezeigt werden kann. Wenn der Benutzer auf das Textfeld klickt, wird ein vertikaler Bildlauf des Textes durchgeführt.

```
package
{
    import flash.display.Sprite;
    import flash.text.*;
    import flash.events.MouseEvent;

    public class TextScrollExample extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myText:String = "Hello world and welcome to the show. It's really nice to
meet you. Take your coat off and stay a while. OK, show is over. Hope you had fun. You can go
home now. Don't forget to tip your waiter. There are mints in the bowl by the door. Thank you.
Please come again.";

        public function TextScrollExample()
        {
            myTextBox.text = myText;
            myTextBox.width = 200;
            myTextBox.height = 50;
            myTextBox.multiline = true;
            myTextBox.wordWrap = true;
            myTextBox.background = true;
            myTextBox.border = true;

            var format:TextFormat = new TextFormat();
            format.font = "Verdana";
            format.color = 0xFF0000;
            format.size = 10;

            myTextBox.defaultTextFormat = format;
            addChild(myTextBox);
            myTextBox.addEventListener(MouseEvent.MOUSE_DOWN, mouseDownScroll);
        }

        public function mouseDownScroll(event:MouseEvent):void
        {
            myTextBox.scrollV++;
        }
    }
}
```

Auswählen und Bearbeiten von Text

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können dynamischen Text oder Eingabetext auswählen. Da bei den Eigenschaften und Methoden der TextField-Klasse zur Textauswahl Indexpositionen verwendet werden, um den Bereich des zu bearbeitenden Textes festzulegen, können Sie dynamischen Text oder Eingabetext im Programmcode auswählen, selbst wenn der Inhalt nicht bekannt ist.

Hinweis: *Wenn Sie in Flash Professional für ein statisches Textfeld die selectable-Option auswählen, handelt es sich bei dem exportierten und in die Anzeigeliste platzierten Textfeld um ein normales dynamisches Textfeld.*

Auswählen von Text

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `flash.text.TextField.selectable`-Eigenschaft hat standardmäßig den Wert `true`. Sie können Text mit Programmanweisungen auswählen, indem Sie die `setSelection()`-Methode verwenden.

Sie können beispielsweise festlegen, dass bestimmter Text in einem Textfeld ausgewählt wird, wenn der Benutzer auf das Textfeld klickt:

```
var myTextField:TextField = new TextField();
myTextField.text = "No matter where you click on this text field the TEXT IN ALL CAPS is selected.";
myTextField.autoSize = TextFieldAutoSize.LEFT;
addChild(myTextField);
addEventListener(MouseEvent.CLICK, selectText);

function selectText(event:MouseEvent):void
{
    myTextField.setSelection(49, 65);
}
```

Wenn Sie festlegen möchten, dass spezifischer Text in einem Textfeld bei der Textanzeige ausgewählt wird, erstellen Sie eine Ereignisprozedurfunktion, die aufgerufen wird, wenn das Textfeld zur Anzeigeliste hinzugefügt wird.

Erfassen des vom Benutzer ausgewählten Textes

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den Eigenschaften `selectionBeginIndex` und `selectionEndIndex` der TextField-Klasse kann der vom Benutzer derzeit ausgewählte Text erfasst werden. Diese Eigenschaften sind schreibgeschützt, sodass Text nicht programmgesteuert ausgewählt werden kann. Zusätzlich kann in Eingabetextfeldern die `caretIndex`-Eigenschaft verwendet werden.

Mit dem folgenden Code werden beispielsweise die Indexpositionswerte des vom Benutzer ausgewählten Textes ausgegeben:

```
var myTextField:TextField = new TextField();
myTextField.text = "Please select the TEXT IN ALL CAPS to see the index values for the first
and last letters.";
myTextField.autoSize = TextFieldAutoSize.LEFT;
addChild(myTextField);
addEventListener(MouseEvent.MOUSE_UP, selectText);

function selectText(event:MouseEvent):void
{
    trace("First letter index position: " + myTextField.selectionBeginIndex);
    trace("Last letter index position: " + myTextField.selectionEndIndex);
}
```

Sie können eine Sammlung von Eigenschaften des TextFormat-Objekts auf die Auswahl anwenden, um die Darstellung des Textes zu ändern. Weitere Informationen zum Anwenden einer Gruppe von TextFormat-Eigenschaften auf ausgewählten Text finden Sie unter „Formatieren von Textbereichen innerhalb eines Textfelds“ auf Seite 410.

Erfassen von Texteingaben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Eigenschaft `type` eines Textfeldes ist standardmäßig auf `dynamic` festgelegt. Wenn Sie mithilfe der TextFieldType-Klasse für die `type`-Eigenschaft den Wert `input` festlegen, können Sie die Benutzereingabe erfassen und den Wert zur Verwendung in anderen Teilen der Anwendung speichern. Eingabetextfelder sind hilfreich bei Formularen und allen Anwendungen, in denen Benutzer einen Textwert für die Verwendung an anderer Stelle im Programm angeben sollen.

Mit dem folgenden Code wird beispielsweise ein Eingabetextfeld mit dem Namen `myTextBox` erstellt. Wenn der Benutzer Text im Textfeld eingibt, wird das `textInput`-Ereignis ausgelöst. Eine Ereignisprozedur mit der Bezeichnung `textInputCapture` erfasst den eingegebenen Textstring und weist diesen String einer Variablen zu. Der neue Text wird in Flash Player oder AIR in einem anderen Textfeld mit dem Namen `myOutputBox` angezeigt.

```
package
{
    import flash.display.Sprite;
    import flash.display.Stage;
    import flash.text.*;
    import flash.events.*;

    public class CaptureUserInput extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myOutputBox:TextField = new TextField();
        private var myText:String = "Type your text here.";

        public function CaptureUserInput()
        {
            captureText();
        }

        public function captureText():void
        {
            myTextBox.type = TextFieldType.INPUT;
            myTextBox.background = true;
            addChild(myTextBox);
```

```
            myTextBox.text = myText;
            myTextBox.addEventListener(TextEvent.TEXT_INPUT, textInputCapture);
        }

        public function textInputCapture(event:TextEvent):void
        {
            var str:String = myTextBox.text;
            createOutputBox(str);
        }

        public function createOutputBox(str:String):void
        {
            myOutputBox.background = true;
            myOutputBox.x = 200;
            addChild(myOutputBox);
            myOutputBox.text = str;
        }

    }
}
```

Einschränken der Texteingabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Da Eingabetextfelder häufig für Formulare und Dialogfelder in Anwendungen eingesetzt werden, empfiehlt es sich, die Art der Zeichen einzuschränken, die Benutzer im Textfeld eingeben können, oder auch den eingegebenen Text zu verbergen, z. B. bei einem Kennwort. Die flash.text.TextField-Klasse verfügt über eine `displayAsPassword`-Eigenschaft und eine `restrict`-Eigenschaft, mit deren Hilfe Sie Benutzereingaben einschränken können.

Mit der `displayAsPassword`-Eigenschaft wird der Text bei der Eingabe durch den Benutzer verborgen (als eine Folge von Sternchen) angezeigt. Wenn `displayAsPassword` den Wert `true` hat, können die Befehle „Ausschneiden“ und „Kopieren“ sowie die entsprechenden Tastaturbefehle nicht verwendet werden. Wie im folgenden Beispiel dargestellt ist, weisen Sie die `displayAsPassword`-Eigenschaft genauso zu wie die anderen Eigenschaften, z. B. „background“ und „color“:

```
myTextBox.type = TextFieldType.INPUT;
myTextBox.background = true;
myTextBox.displayAsPassword = true;
addChild(myTextBox);
```

Die `restrict`-Eigenschaft ist etwas komplizierter, da Sie angeben müssen, welche Zeichen der Benutzer in einem Eingabetextfeld eingeben darf. Sie können einzelne Buchstaben und Ziffern oder Bereiche von Buchstaben, Ziffern und anderen Zeichen angeben. Mit dem folgenden Code können Benutzer im Textfeld nur Großbuchstaben (und keine Ziffern oder Sonderzeichen) eingeben:

```
myTextBox.restrict = "A-Z";
```

In ActionScript 3.0 dienen Bindestriche zur Angabe von Bereichen und Caretzeichen zum Festlegen ausgeschlossener Zeichen. Weitere Informationen zum Definieren von Einschränkungen für Eingabetextfelder finden Sie im ActionScript 3.0-Referenzhandbuch im Eintrag zur `flash.text.TextField.restrict`-Eigenschaft.

Hinweis: *Wenn Sie die `flash.text.TextField.restrict`-Eigenschaft verwenden, konvertiert die Laufzeitumgebung eingeschränkte Buchstaben automatisch in die zulässige Schreibweise. Wenn Sie die `fl.text.TLFTextField.restrict`-Eigenschaft verwenden (d. h., wenn Sie ein TLF-Textfeld verwenden), ignoriert die Laufzeitumgebung eingeschränkte Buchstaben.*

Formatieren von Text

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für die Formatierung der Textanzeige mit Programmanweisungen stehen mehrere Optionen zur Auswahl. Sie können Eigenschaften direkt für die TextField-Instanz festlegen, beispielsweise die Eigenschaften `TextFIeld.thickness`, `TextField.textColor` und `TextField.textHeight`. Sie können auch den Inhalt des Textfelds mit der `htmlText`-Eigenschaft kennzeichnen und die unterstützten HTML-Tags, wie `b`, `i` und `u`, verwenden. Zudem ist es möglich, TextFormat-Objekte auf Textfelder mit unformatiertem Text oder StyleSheet-Objekte auf Textfelder mit der `htmlText`-Eigenschaft anzuwenden. Mit der Verwendung von TextFormat- und StyleSheet-Objekten erzielen Sie die beste Kontrolle und Konsistenz für das Gesamterscheinungsbild von Text in der Anwendung. Sie können ein TextFormat- oder StyleSheet-Objekt definieren und dann auf einige oder alle Textfelder in der Anwendung anwenden.

Zuweisen von Textformaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der TextFormat-Klasse können Sie verschiedene Eigenschaften für die Textanzeige festlegen und auf den gesamten Inhalt eines TextField-Objekts oder auf einen bestimmten Textbereich anwenden.

Im folgenden Beispiel wird ein TextFormat-Objekt auf ein gesamtes TextField-Objekt und ein zweites TextFormat-Objekt auf einen Textbereich in diesem TextField-Objekt angewendet:

```
var tf:TextField = new TextField();
tf.text = "Hello Hello";

var format1:TextFormat = new TextFormat();
format1.color = 0xFF0000;

var format2:TextFormat = new TextFormat();
format2.font = "Courier";

tf.setTextFormat(format1);
var startRange:uint = 6;
tf.setTextFormat(format2, startRange);

addChild(tf);
```

Die `TextField.setTextFormat()`-Methode wirkt sich nur auf Text aus, der bereits im Textfeld angezeigt wird. Wenn sich der Inhalt im TextField-Objekt ändert, muss in der Anwendung möglicherweise erneut die `TextField.setTextFormat()`-Methode aufgerufen werden, damit die Formatierung neu zugewiesen wird. Sie können zudem die `defaultTextFormat`-Eigenschaft des TextField-Objekts so festlegen, dass das zu verwendende Format für vom Benutzer eingegebenen Text angegeben wird.

Anwenden von Cascading Style Sheets

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Textfelder können entweder unformatierten oder HTML-formatierten Text enthalten. Unformatierter Text wird in der `text`-Eigenschaft der Instanz gespeichert und HTML-Text in der `htmlText`-Eigenschaft.

Mithilfe von CSS-Stylesheets können Sie Textformate definieren, die Sie auf viele verschiedene Textfelder anwenden können. CSS-Stylesheets können im Anwendungscode erstellt oder zur Laufzeit aus einer externen CSS-Datei geladen werden.

Mit der flash.text.StyleSheet-Klasse werden CSS-Stile verarbeitet. Mit der StyleSheet-Klasse wird eine begrenzte Anzahl von CSS-Eigenschaften erkannt. Eine detaillierte Aufstellung der Stileigenschaften, die von der StyleSheet-Klasse unterstützt werden, finden Sie im ActionScript 3.0-Referenzhandbuch im Eintrag zur flash.textStylesheet-Eigenschaft.

Wie im folgenden Beispiel dargestellt ist, können Sie CSS-Stylesheets im Code erstellen und mithilfe eines StyleSheet-Objekts auf HTML-Text anwenden:

```
var style:StyleSheet = new StyleSheet();

var styleObj:Object = new Object();
styleObj.fontSize = "bold";
styleObj.color = "#FF0000";
style.setStyle(".darkRed", styleObj);

var tf:TextField = new TextField();
tf.styleSheet = style;
tf.htmlText = "<span class = 'darkRed'>Red</span> apple";

addChild(tf);
```

Nach dem Erstellen eines StyleSheet-Objekts wird im Codebeispiel ein einfaches Objekt zum Aufnehmen einer Gruppe von Eigenschaften für das Stylesheet erstellt. Anschließend wird die `StyleSheet.setStyle()`-Methode aufgerufen, mit der der neue Stil mit dem Namen „.darkred“ zum Stylesheet hinzugefügt wird. Dann wird die Formatierung des Stylesheets angewendet, indem das StyleSheet-Objekt zur `styleSheet`-Eigenschaft des TextField-Objekts zugewiesen wird.

Damit die CSS-Stile wirksam werden, sollte das Stylesheet auf das TextField-Objekt angewendet werden, bevor die `htmlText`-Eigenschaft festgelegt wird.

Ein Textfeld, auf das ein Stylesheet angewendet wurde, kann nicht mehr bearbeitet werden. Wenn Sie einem Eingabetextfeld ein Stylesheet zuweisen, wird das Textfeld mit den Eigenschaften des Stylesheets angezeigt. Benutzer können jedoch keinen neuen Text in das Textfeld eingeben. Darüber hinaus können auch die folgenden ActionScript-APIs nicht mehr für ein Textfeld mit einem zugewiesenen Stylesheet verwendet werden:

- `TextField.replaceText()`-Methode
- `TextField.replaceSelectedText()`-Methode
- `TextField.defaultTextFormat`-Eigenschaft
- `TextField.setTextFormat()`-Methode

Wenn einem Textfeld ein Stylesheet zugewiesen wurde, die `TextField.styleSheet`-Eigenschaft jedoch später auf den Wert `null` gesetzt wird, werden den Inhalten der Eigenschaften `TextField.text` und `TextField.htmlText` Tags und Attribute hinzugefügt, um die Formatierung des zuvor zugewiesenen Stylesheets einzubinden. Um die ursprüngliche `htmlText`-Eigenschaft beizubehalten, müssen Sie diese in einer Variablen speichern, bevor Sie dem Stylesheet den Wert `null` zuweisen.

Laden externer CSS-Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Verwendung von CSS für das Formatieren ist effizienter, wenn Sie CSS-Daten zur Laufzeit aus einer externen Datei laden können. Wenn es sich um externe CSS-Daten außerhalb der Anwendung handelt, können Sie den Anzeigestil des Textes in der Anwendung ändern, ohne Änderungen am ActionScript 3.0-Quellcode vornehmen zu müssen. Nach Bereitstellung der Anwendung können Sie eine externe CSS-Datei so ändern, dass sich das Erscheinungsbild der Anwendung ändert, ohne dass die SWF-Datei der Anwendung neu bereitgestellt werden muss.

Mit der `StyleSheet.parseCSS()`-Methode wird ein String mit CSS-Daten in Stylesheets im StyleSheet-Objekt konvertiert. Im folgenden Beispiel wird veranschaulicht, wie eine externe CSS-Datei abgerufen wird und wie die zugehörigen Stylesheets auf ein TextField-Objekt angewendet werden.

Im Folgenden wird zunächst der Inhalt der zu ladenden CSS-Datei mit dem Namen „example.css“ angezeigt:

```
p {
    font-family: Times New Roman, Times, _serif;
    font-size: 14;
}

h1 {
    font-family: Arial, Helvetica, _sans;
    font-size: 20;
    font-weight: bold;
}

.bluetext {
    color: #0000CC;
}
```

Anschließend folgt der ActionScript-Code für eine Klasse, mit dem die Datei „example.css“ geladen wird und die entsprechenden Stile auf den TextField-Inhalt angewendet werden:

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLLoader;
    import flash.net.URLRequest;
    import flash.text.StyleSheet;
    import flash.text.TextField;
    import flash.text.TextFieldAutoSize;

    public class CSSFormattingExample extends Sprite
    {
        var loader:URLLoader;
        var field:TextField;
        var exampleText:String = "<h1>This is a headline</h1>" +
            "<p>This is a line of text. <span class='bluetext'>" +
            "This line of text is colored blue.</span></p>";

        public function CSSFormattingExample():void
        {
            field = new TextField();
            field.width = 300;
            field.autoSize = TextFieldAutoSize.LEFT;
            field.wordWrap = true;
            addChild(field);

            var req:URLRequest = new URLRequest("example.css");

            loader = new URLLoader();
            loader.addEventListener(Event.COMPLETE, onCSSFileLoaded);
            loader.load(req);
        }

        public function onCSSFileLoaded(event:Event):void
        {
            var sheet:StyleSheet = new StyleSheet();
            sheet.parseCSS(loader.data);
            field.styleSheet = sheet;
            field.htmlText = exampleText;
        }
    }
}
```

Nach dem Laden der CSS-Daten wird mithilfe der `onCSSFileLoaded()`-Methode die `StyleSheet.parseCSS()`-Methode aufgerufen, damit die Stylesheets auf das StyleSheet-Objekt übertragen werden.

Formatieren von Textbereichen innerhalb eines Textfelds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei der `setTextFormat()`-Methode handelt es sich um eine nützliche Methode der flash.text.TextField-Klasse. Mithilfe der `setTextFormat()`-Methode können Sie dem Inhalt eines bestimmten Bereichs des Textfeldes spezifische Eigenschaften zuweisen, um auf Benutzereingaben zu reagieren, z. B. bei Formularen, bei denen Benutzer daran erinnert werden müssen, dass bestimmte Eingaben erforderlich sind, oder um die Darstellung eines Unterabschnitts einer Textpassage in einem Textfeld zu ändern, wenn der Benutzer Textbereiche auswählt.

Im folgenden Beispiel wird die `TextField.setTextFormat()`-Methode auf einen Zeichenbereich angewendet, um die Darstellung eines Teils des Inhalts von `myTextField` zu ändern, wenn der Benutzer auf das Textfeld klickt:

```
var myTextField:TextField = new TextField();
myTextField.text = "No matter where you click on this text field the TEXT IN ALL CAPS changes
format.";
myTextField.autoSize = TextFieldAutoSize.LEFT;
addChild(myTextField);
addEventListener(MouseEvent.CLICK, changeText);

var myformat:TextFormat = new TextFormat();
myformat.color = 0xFF0000;
myformat.size = 18;
myformat.underline = true;

function changeText(event:MouseEvent):void
{
    myTextField.setTextFormat(myformat, 49, 65);
}
```

Erweiterte Textdarstellung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 stellt im flash.text-Paket eine Vielzahl von Klassen zum Festlegen der Eigenschaften von Anzeigetext bereit. Dazu zählen eingebettete Schriftarten, Anti-Aliasing-Einstellungen, Alphakanalsteuerung und weitere spezielle Einstellungen. Im ActionScript 3.0-Referenzhandbuch finden Sie ausführliche Beschreibungen dieser Klassen und Eigenschaften, einschließlich der CSMSettings-, Font- und TextRenderer-Klassen.

Verwenden eingebetteter Schriftarten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie eine bestimmte Schriftart für ein TextField-Objekt in der Anwendung angeben, wird in Flash Player oder AIR eine Geräteschriftart (eine auf dem Computer des Benutzers gespeicherte Schriftart) mit demselben Namen gesucht. Wenn diese Schriftart auf dem Computer des Benutzers nicht gefunden wird oder der Benutzer über eine geringfügig abweichende Schriftartversion mit diesem Namen verfügt, kann sich die Textanzeige erheblich von der gewünschten Anzeige unterscheiden. Standardmäßig wird der Text in der Schriftart Times Roman angezeigt.

Um sicherzustellen, dass genau die gewünschte Schriftart angezeigt wird, können Sie die Schriftart in die SWF-Datei der Anwendung einbetten. Eingebettete Schriftarten weisen mehrere Vorteile auf:

- Bei Zeichen eingebetteter Schriftarten wird das Anti-Aliasing-Verfahren durchgeführt, sodass die Zeichenkonturen glatter erscheinen, vor allem bei großer Schrift.
- Text mit eingebetteten Schriftarten kann gedreht werden.
- Text mit eingebetteten Schriftarten kann transparent oder halbtransparent dargestellt werden.
- Bei eingebetteten Schriftarten kann der CSS-Stil `kerning` verwendet werden.

Der größte Nachteil bei der Verwendung eingebetteter Schriftarten liegt darin, dass sich die Dateigröße oder Downloadgröße der Anwendung erhöht.

Das genaue Verfahren zum Einbetten einer Schriftartdatei in die SWF-Datei der Anwendung hängt von der jeweiligen Entwicklungsumgebung ab.

Nach dem Einbetten einer Schriftart können Sie folgendermaßen überprüfen, ob in einem TextField-Objekt die korrekte eingebettete Schriftart verwendet wird:

- Setzen Sie die `embedFonts`-Eigenschaft des TextField-Objekts auf `true`.
- Erstellen Sie ein TextFormat-Objekt, setzen Sie die zugehörige `fontFamily`-Eigenschaft auf den Namen der eingebetteten Schriftart und wenden Sie das TextFormat-Objekt auf das TextField-Objekt an. Bei der Angabe einer eingebetteten Schriftart sollte die `fontFamily`-Eigenschaft nur einen Namen enthalten. Es können nicht mehrere durch Kommas getrennte Schriftartnamen angegeben werden.
- Wenn Sie Schriftarten für TextField-Objekte oder Komponenten mithilfe von CSS-Stilen festlegen möchten, setzen Sie die CSS-Eigenschaft `font-family` auf den Namen der eingebetteten Schriftart. Die `font-family`-Eigenschaft darf nur einen einzelnen Namen und keine Liste mit Namen enthalten, wenn Sie eine eingebettete Schriftart angeben möchten.

Einbetten von Schriftarten in Flash

In Flash Professional können Sie nahezu alle Schriftarten einbetten, die auf Ihrem Computer installiert sind, einschließlich TrueType-Schriftarten und PostScript Type 1-Schriftarten.

Schriftarten können auf verschiedene Weise in eine Anwendung eingebettet werden, z. B.:

- Festlegen der Schriftart- und Stileigenschaften eines TextField-Objekts auf der Bühne sowie Aktivieren des Kontrollkästchens „Schriftarten einbetten“
- Erstellen eines Schriftartsymbols und Verweisen auf dieses Schriftartsymbol
- Erstellen und Verwenden einer gemeinsam genutzten Laufzeitbibliothek mit eingebetteten Schriftartsymbolen

Weitere Informationen zum Einbetten von Schriftarten in Flash-Anwendungen finden Sie unter „Eingebettete Schriftarten für dynamische Textfelder oder Eingabetextfelder“ im Handbuch *Verwenden von Flash*.

Einbetten von Schriftarten in Flex

Schriftarten können auf verschiedene Weise in eine Flex-Anwendung eingebettet werden, z. B.:

- Verwenden des Metadaten-Tags `[Embed]` in einem Skript
- Verwenden des Stylesheets `@font-face`
- Festlegen einer Klasse für die Schriftart und Einbetten mithilfe des `[Embed]`-Tags

Sie können nur TrueType-Schriftarten direkt in eine Flex-Anwendung einbetten. Schriftarten in anderen Formaten, wie Type 1 Postscript-Schriftarten, müssen zunächst mit Flash Professional in eine SWF-Datei eingebettet werden. Anschließend kann diese SWF-Datei dann in der Flex-Anwendung verwendet werden. Weitere Informationen zur Verwendung eingebetteter Schriftarten aus SWF-Dateien in Flex finden Sie im Handbuch *Using Flex 4* unter „Embedding fonts from SWF files“.

Verwandte Hilfethemen

Einbetten von Schriftarten, um ein konsistentes Textbild zu erhalten

Peter deHaan: Embedding fonts

Divillysausages.com: AS3 Font embedding masterclass

Festlegen der Textschärfe, Schriftstärke und des Anti-Aliasings

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Standardeinstellung werden in Flash Player oder AIR die Einstellungen für die Textanzeige, wie Textschärfe, Schriftstärke und Anti-Aliasing, festgelegt, wenn die Textgröße oder die Farbe geändert oder der Text mit einem anderen Hintergrund angezeigt wird. In einigen Fällen, etwa bei sehr kleinem oder sehr großem Text oder Text mit einer Auswahl besonderer Hintergründe, ist es ggf. wünschenswert, diese Einstellungen selbst zu steuern. Sie können die Flash Player- bzw. AIR-Einstellungen mithilfe der `flash.text.TextRenderer`-Klasse und der entsprechenden zugeordneten Klassen, z. B. der CSMSettings-Klasse, außer Kraft setzen. Mit diesen Klassen können Sie die Darstellungsqualität des eingebetteten Textes genau festlegen. Weitere Informationen zu eingebetteten Schriftarten finden Sie unter „Verwenden eingebetteter Schriftarten“ auf Seite 411.

Hinweis: *Die `flash.text.TextField.antiAliasType`-Eigenschaft muss den Wert `AntiAliasType.ADVANCED` aufweisen, damit Sie die Textschärfe, die Schriftstärke oder die gridFitType-Eigenschaft festlegen oder die `TextRenderer.setAdvancedAntiAliasingTable()`-Methode verwenden können.*

Im folgenden Beispiel werden benutzerdefinierte CSM-Eigenschaften (Continuous Stroke Modulation) und CSM-Formatierungen mit einer eingebetteten Schriftart mit dem Namen `myFont` auf den angezeigten Text angewendet. Wenn der Benutzer auf den angezeigten Text klickt, werden in Flash Player oder Adobe AIR die benutzerdefinierten Einstellungen angewendet:

```
var format:TextFormat = new TextFormat();
format.color = 0x336699;
format.size = 48;
format.font = "myFont";

var myText:TextField = new TextField();
myText.embedFonts = true;
myText.autoSize = TextFieldAutoSize.LEFT;
myText.antiAliasType = AntiAliasType.ADVANCED;
myText.defaultTextFormat = format;
myText.selectable = false;
myText.mouseEnabled = true;
myText.text = "Hello World";
addChild(myText);
myText.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:Event):void
{
    var myAntiAliasSettings = new CSMSettings(48, 0.8, -0.8);
    var myAliasTable:Array = new Array(myAntiAliasSettings);
    TextRenderer.setAdvancedAntiAliasingTable("myFont", FontStyle.ITALIC,
TextColorType.DARK_COLOR, myAliasTable);
}
```

Verwenden von statischem Text

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Statischer Text wird nur in Flash Professional erstellt. Sie können statischen Text nicht im ActionScript-Programmcode instanziieren. Statischer Text ist sinnvoll, wenn es sich um sehr kurzen Text handelt, der sich nicht (wie dynamischer Text) ändern soll. Stellen Sie sich statischen Text als eine Art Grafikelement vor, z. B. ein Kreis oder ein Rechteck, das mit Flash Professional auf der Bühne gezeichnet wird. Zwar unterliegt statischer Text mehr Einschränkungen als dynamischer Text, ActionScript 3.0 unterstützt jedoch die Möglichkeit, die Eigenschaftswerte für statischen Text mithilfe der StaticText-Klasse zu lesen. Darüber hinaus können Sie mit der TextSnapshot-Klasse Werte aus dem statischen Text lesen.

Zugreifen auf statische Textfelder mit der StaticText-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Regel verwenden Sie für die Interaktion mit einer auf der Bühne platzierten statischen Textinstanz die flash.text.StaticText-Klasse im Bedienfeld „Aktionen“ von Flash Professional. Sie können auch ActionScript-Dateien verwenden, die mit einer SWF-Datei interagieren, die statischen Text enthält. In beiden Fällen ist es nicht möglich, eine statische Textinstanz mit Programmanweisungen zu instanziieren. Statischer Text wird in Flash Professional erstellt.

Wenn Sie einen Verweis auf ein vorhandenes statisches Textfeld erstellen möchten, können Sie die Elemente in der Anzeigeliste durchlaufen und ihnen eine Variable zuweisen. Zum Beispiel:

```
for (var i = 0; i < this.numChildren; i++) {
var displayitem:DisplayObject = this.getChildAt(i);
if (displayitem instanceof StaticText) {
trace("a static text field is item " + i + " on the display list");
        var myFieldLabel:StaticText = StaticText(displayitem);
        trace("and contains the text: " + myFieldLabel.text);
}
}
```

Wenn Sie über einen Verweis auf ein statisches Textfeld verfügen, können Sie die Eigenschaften dieses Feldes in ActionScript 3.0 verwenden. Der folgende Code wird mit einem Bild in der Zeitleiste verknüpft und setzt voraus, dass dem Verweis auf den statischen Text eine Variable mit dem Namen `myFieldLabel` zugewiesen wurde. Ein dynamisches Textfeld mit dem Namen `myField` wird relativ zum `x`- und `y`-Wert von `myFieldLabel` positioniert und zeigt den Wert von `myFieldLabel` erneut an.

```
var myField:TextField = new TextField();
addChild(myField);
myField.x = myFieldLabel.x;
myField.y = myFieldLabel.y + 20;
myField.autoSize = TextFieldAutoSize.LEFT;
myField.text = "and " + myFieldLabel.text
```

Verwenden der TextSnapshot-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie eine vorhandene statische Textinstanz im Programmcode verwenden möchten, können Sie mithilfe der flash.text.TextSnapshot-Klasse die `textSnapshot`-Eigenschaft eines flash.display.DisplayObjectContainer-Objekts auslesen. Anders ausgedrückt, Sie erstellen eine TextSnapshot-Instanz der `DisplayObjectContainer.textSnapshot`-Eigenschaft. Sie können dann auf diese Instanz Methoden anwenden, um Werte abzurufen oder Teile des statischen Textes auszuwählen.

Platzieren Sie beispielsweise ein statisches Textfeld mit dem Text „TextSnapshot Example“ auf der Bühne. Fügen Sie in Bild 1 in der Zeitleiste den folgenden ActionScript-Code hinzu:

```
var mySnap:TextSnapshot = this.textSnapshot;
var count:Number = mySnap.charCount;
mySnap.setSelected(0, 4, true);
mySnap.setSelected(1, 2, false);
var myText:String = mySnap.getSelectedText(false);
trace(myText);
```

Die TextSnapshot-Klasse ist hilfreich beim Abrufen von Text aus statischen Textfeldern in einer geladenen SWF-Datei, wenn Sie diesen Text als Wert in einem anderen Teil der Anwendung verwenden möchten.

TextField-Beispiel: Textformatierung im Zeitungsstil

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Beispielanwendung „News Layout“ wird Text so formatiert, dass er einem Zeitungsartikel ähnelt. Der Eingabetext kann eine Schlagzeile, einen Untertitel und den Haupttext des Artikels enthalten. In Bezug auf die Breite und Höhe der Anzeige werden in der Beispielanwendung „News Layout“ die Schlagzeile und der Untertitel so formatiert, dass sie in voller Breite im Anzeigebereich dargestellt werden. Der Haupttext wird auf mindestens zwei Spalten aufgeteilt.

In diesem Beispiel werden die folgenden ActionScript-Programmiertechniken vermittelt:

- Erweitern der TextField-Klasse
- Laden und Anwenden einer externen CSS-Datei
- Konvertieren von CSS-Stilen in TextFormat-Objekte
- Abrufen von Daten zur Größe der Textanzeige mithilfe der TextLineMetrics-Klasse

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „News Layout“ befinden sich im Ordner „Samples/NewsLayout“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
NewsLayout.mxml oder NewsLayout.fla	Die Benutzeroberfläche der Anwendung im Flex-Format (MXML) oder Flash-Format (FLA).
com/example/programmingas3/newslayout/StoryLayoutComponent.as	Eine UIComponent-Klasse von Flex, durch die die StoryLayout-Instanz eingefügt wird.
com/example/programmingas3/newslayout/StoryLayout.as	Die ActionScript-Hauptklasse, mit der alle Komponenten des Zeitungsartikels für die Anzeige angeordnet werden.
com/example/programmingas3/newslayout/FormattedTextField.as	Eine Unterklasse der TextField-Klasse, in der das zugehörige TextFormat-Objekt verwaltet wird.
com/example/programmingas3/newslayout/HeadlineTextField.as	Eine Unterklasse der FormattedTextField-Klasse, mit der die Schriftgröße so angepasst wird, dass der Text die gewünschte Breite hat.
com/example/programmingas3/newslayout/MultiColumnTextField.as	Eine ActionScript-Klasse, mit der Text auf mindestens zwei Spalten aufgeteilt wird.
story.css	Eine CSS-Datei, mit der Textformate für das Layout definiert werden.

Laden der externen CSS-Datei

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Anwendung „News Layout“ wird zunächst der Text des Artikels aus einer lokalen XML-Datei geladen. Anschließend wird eine externe CSS-Datei mit Formatierungsdaten für die Schlagzeile, den Untertitel und den Haupttext geladen.

In der CSS-Datei sind drei Stile definiert, ein Standardabsatzformat für den Artikel sowie das h1-Format und das h2-Format für die Schlagzeile bzw. den Untertitel.

```
p {
    font-family: Georgia, "Times New Roman", Times, _serif;
    font-size: 12;
    leading: 2;
    text-align: justify;
    indent: 24;
}

h1 {
    font-family: Verdana, Arial, Helvetica, _sans;
    font-size: 20;
    font-weight: bold;
    color: #000099;
    text-align: left;
}

h2 {
    font-family: Verdana, Arial, Helvetica, _sans;
    font-size: 16;
    font-weight: normal;
    text-align: left;
}
```

Das zum Lesen der externen CSS-Datei verwendete Verfahren entspricht dem Verfahren, das im Abschnitt „Laden externer CSS-Dateien“ auf Seite 409 beschrieben wird. Nach dem Laden der CSS-Datei wird in der Anwendung die `onCSSFileLoaded()`-Methode ausgeführt, wie im Folgenden dargestellt.

```
public function onCSSFileLoaded(event:Event):void
{
    this.sheet = new StyleSheet();
    this.sheet.parseCSS(loader.data);

    h1Format = getTextStyle("h1", this.sheet);
    if (h1Format == null)
    {
        h1Format = getDefaultHeadFormat();
    }
    h2Format = getTextStyle("h2", this.sheet);
    if (h2Format == null)
    {
        h2Format = getDefaultHeadFormat();
        h2Format.size = 16;
    }
    pFormat = getTextStyle("p", this.sheet);
    if (pFormat == null)
    {
        pFormat = getDefaultTextFormat();
        pFormat.size = 12;
    }
    displayText();
}
```

Mit der `onCSSFileLoaded()`-Methode wird ein StyleSheet-Objekt erstellt. Mit diesem Objekt werden dann die CSS-Eingabedaten analysiert. Der Haupttext des Artikels wird in einem MultiColumnTextField-Objekt angezeigt, in dem StyleSheet-Objekte direkt verwendet werden können. Für die Felder der Schlagzeile wird dagegen die HeadlineTextField-Klasse verwendet. Die entsprechende Formatierung erfolgt mithilfe eines TextFormat-Objekts.

Mit der `onCSSFileLoaded()`-Methode wird die `getTextStyle()`-Methode zweimal aufgerufen, um ein CSS-Stylesheet in ein TextFormat-Objekt zu konvertieren, das in jedem der beiden HeadlineTextField-Objekte verwendet wird.

```
public function getTextStyle(styleName:String, ss:StyleSheet):TextFormat
{
    var format:TextFormat = null;

    var style:Object = ss.getStyle(styleName);
    if (style != null)
    {
        var colorStr:String = style.color;
        if (colorStr != null && colorStr.indexOf("#") == 0)
        {
            style.color = colorStr.substr(1);
        }
        format = new TextFormat(style.fontFamily,
                        style.fontSize,
                        style.color,
                        (style.fontWeight == "bold"),
                        (style.fontStyle == "italic"),
                        (style.textDecoration == "underline"),
                        style.url,
                        style.target,
                        style.textAlign,
                        style.marginLeft,
                        style.marginRight,
                        style.indent,
                        style.leading);

        if (style.hasOwnProperty("letterSpacing"))
        {
            format.letterSpacing = style.letterSpacing;
        }
    }
    return format;
}
```

Die Eigenschaftsnamen und die Bedeutung der Eigenschaftswerte unterscheiden sich bei CSS-Stylesheets und bei TextFormat-Objekten. Mit der `getTextStyle()`-Methode werden die CSS-Eigenschaftswerte in die im TextFormat-Objekt erwarteten Werte umgewandelt.

Anordnen der Artikelelemente auf der Seite

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der StoryLayout-Klasse werden die Formatierung und das Layout der Felder für Schlagzeile, Untertitel und Haupttext im Zeitungsstil durchgeführt. Mit der `displayText()`-Methode werden die verschiedenen Felder zunächst erstellt und dann positioniert.

```
public function displayText():void
{
    headlineTxt = new HeadlineTextField(h1Format);
    headlineTxt.wordWrap = true;
    headlineTxt.x = this.paddingLeft;
    headlineTxt.y = this.paddingTop;
    headlineTxt.width = this.preferredWidth;
    this.addChild(headlineTxt);

    headlineTxt.fitText(this.headline, 1, true);

    subtitleTxt = new HeadlineTextField(h2Format);
    subtitleTxt.wordWrap = true;
    subtitleTxt.x = this.paddingLeft;
    subtitleTxt.y = headlineTxt.y + headlineTxt.height;
    subtitleTxt.width = this.preferredWidth;
    this.addChild(subtitleTxt);

    subtitleTxt.fitText(this.subtitle, 2, false);

    storyTxt = new MultiColumnText(this.numColumns, 20,
                        this.preferredWidth, 400, true, this.pFormat);
    storyTxt.x = this.paddingLeft;
    storyTxt.y = subtitleTxt.y + subtitleTxt.height + 10;
    this.addChild(storyTxt);

    storyTxt.text = this.content;
...
```

Die einzelnen Felder werden jeweils unter dem vorherigen Feld platziert, indem für die `y`-Eigenschaft des jeweiligen Feldes der gleiche Wert wie für die `y`-Eigenschaft des vorherigen Feldes zuzüglich der Feldhöhe festgelegt wird. Diese dynamische Positionierungsberechnung ist erforderlich, da sich die Größe von HeadlineTextField-Objekten und MultiColumnTextField-Objekten entsprechend der Höhe des jeweiligen Inhalts ändern kann.

Ändern der Schriftgröße entsprechend der Feldgröße

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Entsprechend der angegebenen Breite in Pixel und einer maximalen Anzahl anzuzeigender Zeilen wird die Schriftgröße im HeadlineTextField-Objekt so geändert, dass Text und Feld angepasst werden. Bei einem kurzen Text ist die Schrift groß, sodass eine Schlagzeile im Boulevardstil erstellt wird. Je länger der Text ist, umso kleiner ist die Schriftgröße.

Die Schriftgröße wird mit der `HeadlineTextField.fitText()`-Methode geändert, wie im Folgenden dargestellt:

```
public function fitText(msg:String, maxLines:uint = 1, toUpper:Boolean = false,
targetWidth:Number = -1):uint
{
    this.text = toUpper ? msg.toUpperCase() : msg;

    if (targetWidth == -1)
    {
        targetWidth = this.width;
    }

    var pixelsPerChar:Number = targetWidth / msg.length;

    var pointSize:Number = Math.min(MAX_POINT_SIZE, Math.round(pixelsPerChar * 1.8 * maxLines));

    if (pointSize < 6)
    {
        // the point size is too small
        return pointSize;
    }

    this.changeSize(pointSize);

    if (this.numLines > maxLines)
    {
        return shrinkText(--pointSize, maxLines);
    }
    else
    {
        return growText(pointSize, maxLines);
    }
}

public function growText(pointSize:Number, maxLines:uint = 1):Number
{
    if (pointSize >= MAX_POINT_SIZE)
    {
        return pointSize;
    }

    this.changeSize(pointSize + 1);

    if (this.numLines > maxLines)
    {
        // set it back to the last size
        this.changeSize(pointSize);
        return pointSize;
    }
    else
    {
        return growText(pointSize + 1, maxLines);
    }
```

```
}

public function shrinkText(pointSize:Number, maxLines:uint=1):Number
{
    if (pointSize <= MIN_POINT_SIZE)
    {
        return pointSize;
    }

    this.changeSize(pointSize);

    if (this.numLines > maxLines)
    {
        return shrinkText(pointSize - 1, maxLines);
    }
    else
    {
        return pointSize;
    }
}
```

In der `HeadlineTextField.fitText()`-Methode wird ein einfaches rekursives Verfahren zum Ändern der Schriftgröße verwendet. Zunächst wird die durchschnittliche Anzahl von Pixeln pro Zeichen im Text geschätzt und davon ausgehend eine anfängliche Punktgröße berechnet. Dann wird die Schriftgröße geändert und überprüft, ob im Text Zeilenumbrüche aufgetreten sind und eine die maximale Anzahl zulässiger Textzeilen übersteigende Anzahl von Zeilen entstanden ist. Wenn zu viele Zeilen vorhanden sind, wird die `shrinkText()`-Methode aufgerufen, um die Schriftgröße zu verringern. Anschließend wird der Vorgang wiederholt. Wenn die maximale Anzahl zulässiger Textzeilen nicht überschritten wurde, wird die `growText()`-Methode aufgerufen, um die Schriftgröße zu vergrößern. Dann wird der Vorgang wiederholt. Der Vorgang wird beendet, wenn durch Inkrementieren der Schriftgröße um einen weiteren Punkt zu viele Zeilen entstehen.

Aufteilen von Text auf mehrere Spalten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der MultiColumnTextField-Klasse wird Text auf mehrere TextField-Objekte aufgeteilt, die dann wie Zeitungsspalten angeordnet werden.

Mit dem `MultiColumnTextField()`-Konstruktor wird zunächst ein Array von TextField-Objekten erstellt, jeweils ein Objekt pro Zeile, wie im Folgenden dargestellt:

```
for (var i:int = 0; i < cols; i++)
{
    var field:TextField = new TextField();
    field.multiline = true;
    field.autoSize = TextFieldAutoSize.NONE;
    field.wordWrap = true;
    field.width = this.colWidth;
    field.setTextFormat(this.format);
    this.fieldArray.push(field);
    this.addChild(field);
}
```

Jedes TextField-Objekt wird mithilfe der `addChild()`-Methode zum Array und zur Anzeigeliste hinzugefügt.

Wenn sich die `text`-Eigenschaft oder die `styleSheet`-Eigenschaft des StoryLayout-Objekts ändert, wird die `layoutColumns()`-Methode aufgerufen, damit der Text neu angezeigt wird. Mit der `layoutColumns()`-Methode wird die `getOptimalHeight()`-Methode aufgerufen, um die entsprechende Pixelhöhe zu ermitteln, die zum Anpassen des Textes an die angegebene Layoutbreite erforderlich ist.

```
public function getOptimalHeight(str:String):int
{
    if (field.text == "" || field.text == null)
    {
        return this.preferredHeight;
    }
    else
    {
        this.linesPerCol = Math.ceil(field.numLines / this.numColumns);

        var metrics:TextLineMetrics = field.getLineMetrics(0);
        this.lineHeight = metrics.height;
        var prefHeight:int = linesPerCol * this.lineHeight;

        return prefHeight + 4;
    }
}
```

Zunächst wird mithilfe der `getOptimalHeight()`-Methode die Breite jeder Spalte berechnet. Anschließend werden die Breite und die `htmlText`-Eigenschaft des ersten TextField-Objekts im Array festgelegt. Mithilfe der `getOptimalHeight()`-Methode wird anhand des ersten TextField-Objekts die Gesamtanzahl der Zeilen mit Zeilenumbruch im Text festgestellt und dann die Anzahl der in jeder Spalte erforderlichen Zeilen ermittelt. Dann wird die `TextField.getLineMetrics()`-Methode aufgerufen, um ein TextLineMetrics-Objekt mit detaillierten Daten zur Größe des Textes in der ersten Zeile abzurufen. Mit der `TextLineMetrics.height`-Eigenschaft wird die volle Höhe einer Textzeile in Pixel angegeben, einschließlich Oberlänge, Unterlänge und Durchschuss. Die optimale Höhe des MultiColumnTextField-Objekts ergibt sich dann aus der Zeilenhöhe multipliziert mit der Anzahl der Zeilen pro Spalte, plus 4 für die Ränder eines TextField-Objekts von jeweils zwei Pixel oben und unten.

Es folgt der vollständige Code für die `layoutColumns()`-Methode:

```
public function layoutColumns():void
{
    if (this._text == "" || this._text == null)
    {
        return;
    }

    var field:TextField = fieldArray[0] as TextField;
    field.text = this._text;
    field.setTextFormat(this.format);

    this.preferredHeight = this.getOptimalHeight(field);

    var remainder:String = this._text;
    var fieldText:String = "";
    var lastLineEndedPara:Boolean = true;

    var indent:Number = this.format.indent as Number;

    for (var i:int = 0; i < fieldArray.length; i++)
    {
```

```
field = this.fieldArray[i] as TextField;

field.height = this.preferredHeight;
field.text = remainder;

field.setTextFormat(this.format);

var lineLen:int;
if (indent > 0 && !lastLineEndedPara && field.numLines > 0)
{
    lineLen = field.getLineLength(0);
    if (lineLen > 0)
    {
        field.setTextFormat(this.firstLineFormat, 0, lineLen);
        }
    }

field.x = i * (colWidth + gutter);
field.y = 0;

remainder = "";
fieldText = "";

var linesRemaining:int = field.numLines;
var linesVisible:int = Math.min(this.linesPerCol, linesRemaining);

for (var j:int = 0; j < linesRemaining; j++)
{
    if (j < linesVisible)
    {
        fieldText += field.getLineText(j);
    }
    else
    {
        remainder +=field.getLineText(j);
    }
}

field.text = fieldText;

field.setTextFormat(this.format);

if (indent > 0 && !lastLineEndedPara)
{
    lineLen = field.getLineLength(0);
    if (lineLen > 0)
    {
        field.setTextFormat(this.firstLineFormat, 0, lineLen);
    }
}

var lastLine:String = field.getLineText(field.numLines - 1);
var lastCharCode:Number = lastLine.charCodeAt(lastLine.length - 1);
```

```
        if (lastCharCode == 10 || lastCharCode == 13)
        {
        lastLineEndedPara = true;
        }
        else
        {
        lastLineEndedPara = false;
        }

        if ((this.format.align == TextFormatAlign.JUSTIFY) &&
                (i < fieldArray.length - 1))
        {
        if (!lastLineEndedPara)
        {
            justifyLastLine(field, lastLine);
        }
    }
    }
}
```

Nachdem die `preferredHeight`-Eigenschaft durch Aufrufen der `getOptimalHeight()`-Methode festgelegt wurde, werden die TextField-Objekte mithilfe der `layoutColumns()`-Methode durchlaufen und die Höhe jedes Textfelds jeweils auf den `preferredHeight`-Wert gesetzt. Anschließend wird mithilfe der `layoutColumns()`-Methode jedem Textfeld genau die Anzahl von Zeilen zugewiesen, bei der in keinem Textfeld ein Bildlauf durchgeführt werden muss und der Text in jedem Feld jeweils an der Stelle beginnt, an der der Text im vorherigen Feld endet. Wenn als Textausrichtungsstil „justify“ festgelegt wurde, wird anschließend die `justifyLastLine()`-Methode aufgerufen, um die abschließende Textzeile in einem Feld auszurichten. Andernfalls wird die letzte Zeile als Absatzende eingestuft und nicht ausgerichtet.

Kapitel 22: Verwenden der Flash Text Engine

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die ab Flash Player 10 und Adobe® AIR™ 1.5 verfügbare Adobe® Flash® Text Engine bietet eine elementare Unterstützung für anspruchsvolle Steuerungsaufgaben im Hinblick auf Textmetrik, Textformatierung und bidirektionalen Text. Sie bietet zudem einen verbesserten Textfluss und eine erweiterte Sprachunterstützung. Sie können mit der Flash Text Engine zwar auch einfache Textelemente erstellen und verwalten, sie wurde jedoch primär als Grundlage für die Entwicklung von Textverarbeitungskomponenten konzipiert. Die Flash Text Engine setzt somit fortgeschrittene Programmierungserfahrung voraus. Informationen zum Anzeigen einfacher Textelemente finden Sie unter „Verwenden der TextField-Klasse“ auf Seite 399.

Das Text Layout Framework, das eine textverarbeitende Komponente enthält, die auf der FTE basiert, bietet eine einfachere Möglichkeit, deren erweiterte Funktionen zu verwenden. Das Text Layout Framework ist eine erweiterbare Bibliothek, die komplett in ActionScript 3.0 geschrieben wurde. Sie können die vorhandene TLF-Komponente verwenden oder mit dem Framework Ihre eigene Textkomponente erstellen. Weitere Informationen finden Sie unter „Verwenden des Text Layout Framework“ auf Seite 455.

Verwandte Hilfethemen

flash.text.engine-Paket

Erstellen und Anzeigen von Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Klassen in der Flash Text Engine ermöglichen Ihnen, Text zu erstellen, zu formatieren und zu steuern. Die folgenden Klassen stellen die Grundbausteine für das Erstellen und Anzeigen von Text mit der Flash Text Engine dar:

- TextElement/GraphicElement/GroupElement – Diese Klassen enthalten den Inhalt einer TextBlock-Instanz.
- ElementFormat – Diese Klasse gibt Formatierungsattribute für den Inhalt einer TextBlock-Instanz an.
- TextBlock – Diese Klasse dient zum Erstellen eines Textabsatzes.
- TextLine – Dies ist eine aus der TextBlock-Klasse erstellte Textzeile.

Um Text anzuzeigen, erstellen Sie ein TextElement-Objekt aus einem String. Die Formatierungsmerkmale legen Sie dabei mit einem ElementFormat-Objekt fest. Weisen Sie das TextElement der `content`-Eigenschaft eines TextBlocks zu. Erstellen Sie die anzuzeigenden Textzeilen, indem Sie die `TextBlock.createTextLine()`-Methode aufrufen. Die `createTextLine()`-Methode gibt ein TextLine-Objekt zurück, das den Teil des Strings enthält, der in die angegebene Breite passt. Rufen Sie die Methode wiederholt auf, bis der gesamte String in Zeilen formatiert wurde. Wenn keine weiteren Zeilen erstellt werden müssen, wird der textLineCreationResult-Eigenschaft des TextBlock-Objekts der folgende Wert zugewiesen: `TextLineCreationResult.COMPLETE`. Um die Zeilen anzuzeigen, fügen Sie sie der Anzeigeliste hinzu (mit den entsprechenden Werten für die `x`- und `y`-Position).

Der folgende Code verwendet beispielsweise diese FTE-Klassen, um „Hello World! This is Flash Text Engine!“ mit dem Standardformat und den standardmäßigen Schriftartwerten anzuzeigen. In diesem einfachen Beispiel wird nur eine einzelne Textzeile erstellt.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class HelloWorldExample extends Sprite
    {
        public function HelloWorldExample()
        {
            var str = "Hello World! This is Flash Text Engine!";
            var format:ElementFormat = new ElementFormat();
            var textElement:TextElement = new TextElement(str, format);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = textElement;

            var textLine1:TextLine = textBlock.createTextLine(null, 300);
            addChild(textLine1);
            textLine1.x = 30;
            textLine1.y = 30;
        }
    }
}
```

Die Parameter für `createTextLine()` geben die Zeile an, ab der mit einer neuen Zeile begonnen wird, sowie die Breite der Zeile in Pixel. Die neue Zeile wird im Allgemeinen bei der vorherigen Zeile begonnen, im Falle der ersten Zeile ist der Zeilenanfang jedoch `null`.

Hinzufügen von GraphicElement- und GroupElement-Objekten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können ein GraphicElement-Objekt zu einem TextBlock-Objekt zuweisen, um ein Bild oder ein grafisches Element anzuzeigen. Erstellen Sie hierzu einfach eine Instanz der GraphicElement-Klasse aus einer Grafik oder einem Bild, und weisen Sie der `TextBlock.content`-Eigenschaft diese Instanz zu. Erstellen Sie die Textzeile, indem Sie `TextBlock.createTextline()` wie gewohnt aufrufen. Im folgenden Beispiel werden zwei Textzeilen erstellt, eine mit einem GraphicElement-Objekt und eine mit einem TextElement-Objekt.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.display.Graphics;

    public class GraphicElementExample extends Sprite
    {
        public function GraphicElementExample()
        {
            var str:String = "Beware of Dog!";

            var triangle:Shape = new Shape();
            triangle.graphics.beginFill(0xFF0000, 1);
            triangle.graphics.lineStyle(3);
            triangle.graphics.moveTo(30, 0);
            triangle.graphics.lineTo(60, 50);
            triangle.graphics.lineTo(0, 50);
            triangle.graphics.lineTo(30, 0);
            triangle.graphics.endFill();

            var format:ElementFormat = new ElementFormat();
            format.fontSize = 20;

            var graphicElement:GraphicElement = new GraphicElement(triangle, triangle.width,
triangle.height, format);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = graphicElement;
            var textLine1:TextLine = textBlock.createTextLine(null, triangle.width);
            textLine1.x = 50;
            textLine1.y = 110;
            addChild(textLine1);

            var textElement:TextElement = new TextElement(str, format);
            textBlock.content = textElement;
            var textLine2 = textBlock.createTextLine(null, 300);
            addChild(textLine2);
            textLine2.x = textLine1.x - 30;
            textLine2.y = textLine1.y + 15;
        }
    }
}
```

Sie können ein GroupElement-Objekt erstellen, um eine Gruppe von TextElement-, GraphicElement- oder anderen GroupElement-Objekten zu erstellen. Ein GroupElement kann der `content`-Eigenschaft eines TextBlock-Objekts zugewiesen werden. Der Parameter für den `GroupElement()`-Konstruktor ist ein Vektor, der auf die Text-, Grafik- und Gruppenelemente verweist, aus denen die Gruppe zusammengesetzt ist. Im folgenden Beispiel werden zwei grafische Elemente und ein Textelement gruppiert und als eine Einheit zu einem Textblock zugewiesen.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.display.Graphics;

    public class GroupElementExample extends Sprite
    {
        public function GroupElementExample()
        {
            var str:String = "Beware of Alligators!";

            var triangle1:Shape = new Shape();
            triangle1.graphics.beginFill(0xFF0000, 1);
            triangle1.graphics.lineStyle(3);
            triangle1.graphics.moveTo(30, 0);
            triangle1.graphics.lineTo(60, 50);
            triangle1.graphics.lineTo(0, 50);
            triangle1.graphics.lineTo(30, 0);
            triangle1.graphics.endFill();

            var triangle2:Shape = new Shape();
            triangle2.graphics.beginFill(0xFF0000, 1);
            triangle2.graphics.lineStyle(3);
            triangle2.graphics.moveTo(30, 0);
            triangle2.graphics.lineTo(60, 50);
            triangle2.graphics.lineTo(0, 50);
            triangle2.graphics.lineTo(30, 0);
            triangle2.graphics.endFill();

            var format:ElementFormat = new ElementFormat();
            format.fontSize = 20;
            var graphicElement1:GraphicElement = new GraphicElement(triangle1,
triangle1.width, triangle1.height, format);
            var textElement:TextElement = new TextElement(str, format);
            var graphicElement2:GraphicElement = new GraphicElement(triangle2,
triangle2.width, triangle2.height, format);
            var groupVector:Vector.<ContentElement> = new Vector.<ContentElement>();
            groupVector.push(graphicElement1, textElement, graphicElement2);
            var groupElement = new GroupElement(groupVector);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = groupElement;
            var textLine:TextLine = textBlock.createTextLine(null, 800);
            addChild(textLine);
            textLine.x = 100;
            textLine.y = 200;
        }
    }
}
```

Ersetzen von Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können Text in einer TextBlock-Instanz ersetzen, indem Sie die `TextElement.replaceText()`-Methode aufrufen. Hierdurch wird Text in dem TextElement-Objekt ersetzt, das Sie der `TextBlock.content`-Eigenschaft zugewiesen haben.

Im folgenden Beispiel wird die Methode `replaceText()` zunächst zum Einfügen von Text am Anfang der Zeile, dann zum Anhängen von Text am Ende der Zeile und schließlich zum Ersetzen von Text in der Mitte der Zeile verwendet.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class ReplaceTextExample extends Sprite
    {
        public function ReplaceTextExample()
        {

            var str:String = "Lorem ipsum dolor sit amet";
            var fontDescription:FontDescription = new FontDescription("Arial");
            var format:ElementFormat = new ElementFormat(fontDescription);
            format.fontSize = 14;
            var textElement:TextElement = new TextElement(str, format);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = textElement;
            createLine(textBlock, 10);
            textElement.replaceText(0, 0, "A text fragment: ");
            createLine(textBlock, 30);
            textElement.replaceText(43, 43, "...");
            createLine(textBlock, 50);
            textElement.replaceText(23, 28, "(ipsum)");
            createLine(textBlock, 70);
        }

        function createLine(textBlock:TextBlock, y:Number):void {
            var textLine:TextLine = textBlock.createTextLine(null, 300);
            textLine.x = 10;
            textLine.y = y;
            addChild(textLine);
        }
    }
}
```

Mit der Methode `replaceText()` wird der durch die Parameter `beginIndex` und `endIndex` angegebene Text durch den Text ersetzt, der durch den Parameter `newText` angegeben wird. Die Werte der Parameter `beginIndex` und `endIndex` sind gleich, und die Methode `replaceText()` fügt den Text an dieser Position ein. Andernfalls würden die Zeichen, die durch die Parameter `beginIndex` und `endIndex` angegeben werden, durch den neuen Text ersetzt.

Verarbeiten von Ereignissen in FTE

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können einer TextLine-Instanz Ereignis-Listener hinzufügen, ebenso wie dies bei anderen Anzeigeobjekten möglich ist. So können Sie beispielsweise erkennen, wenn ein Benutzer die Maus über eine Textzeile bewegt oder auf eine Zeile klickt. Im folgenden Beispiel werden beide Ereignisse erkannt. Wenn Sie die Maus über die Zeile bewegen, ändert sich der Cursor in einen Schaltflächen-Cursor, und wenn Sie auf die Zeile klicken, ändert sich die Farbe.

```
package
{
    import flash.text.engine.*;
    import flash.ui.Mouse;
    import flash.display.Sprite
    import flash.events.MouseEvent;
    import flash.events.EventDispatcher;

    public class EventHandlerExample extends Sprite
    {
        var textBlock:TextBlock = new TextBlock();

        public function EventHandlerExample():void
        {
            var str:String = "I'll change color if you click me.";
            var fontDescription:FontDescription = new FontDescription("Arial");
            var format:ElementFormat = new ElementFormat(fontDescription, 18);
            var textElement = new TextElement(str, format);
            textBlock.content = textElement;
            createLine(textBlock);
        }

        private function createLine(textBlock:TextBlock):void
        {
            var textLine:TextLine = textBlock.createTextLine(null, 500);
            textLine.x = 30;
            textLine.y = 30;
            addChild(textLine);
            textLine.addEventListener("mouseOut", mouseOutHandler);
            textLine.addEventListener("mouseOver", mouseOverHandler);
            textLine.addEventListener("click", clickHandler);
        }

        private function mouseOverHandler(event:MouseEvent):void
        {
            Mouse.cursor = "button";
        }

        private function mouseOutHandler(event:MouseEvent):void
        {
            Mouse.cursor = "arrow";
        }

        function clickHandler(event:MouseEvent):void {
            if(textBlock.firstLine)
                removeChild(textBlock.firstLine);
```

```
            var newFormat:ElementFormat = textBlock.content.elementFormat.clone();
            switch(newFormat.color)
            {
                case 0x000000:
                    newFormat.color = 0xFF0000;
                    break;
                case 0xFF0000:
                    newFormat.color = 0x00FF00;
                    break;
                case 0x00FF00:
                    newFormat.color = 0x0000FF;
                    break;
                case 0x0000FF:
                    newFormat.color = 0x000000;
                    break;
            }
            textBlock.content.elementFormat = newFormat;
            createLine(textBlock);
        }
    }
}
```

Spiegeln von Ereignissen

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Darüber hinaus können Sie Ereignisse in einem Textblock oder einem Teil eines Textblocks auf einem Ereignis-Dispatcher spiegeln. Erstellen Sie hierzu zunächst eine EventDispatcher-Instanz, und weisen Sie der Eigenschaft `eventMirror` einer TextElement-Instanz diese Instanz zu. Wenn der Textblock aus einem einzelnen Textelement besteht, spiegelt die Flash Text Engine Ereignisse für den gesamten Block. Besteht der Block aus mehreren Textelementen, spiegelt die Flash Text Engine Ereignisse nur für die TextElement-Instanzen, bei denen die Eigenschaft `eventMirror` festgelegt ist. Der Text im folgenden Beispiel besteht aus drei Elementen: dem Wort „Click“, dem Wort „here“ und dem String „to see me in italic“. In dem Beispiel wird dem zweiten Textelement, dem Wort „here“, ein Ereignis-Dispatcher zugewiesen, und es wird ein Ereignis-Listener, die Methode `clickHandler()`, hinzugefügt. Die `clickHandler()`-Methode ändert die Formatierung des Textes in kursiv. Darüber hinaus wird der Inhalt des dritten Textelements in „Click here to see me in normal font!“ geändert.

```
package
{
    import flash.text.engine.*;
    import flash.ui.Mouse;
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.events.EventDispatcher;

    public class EventMirrorExample extends Sprite
    {
        var fontDescription:FontDescription = new FontDescription("Helvetica", "bold");
        var format:ElementFormat = new ElementFormat(fontDescription, 18);
        var textElement1 = new TextElement("Click ", format);
        var textElement2 = new TextElement("here ", format);
        var textElement3 = new TextElement("to see me in italic! ", format);
        var textBlock:TextBlock = new TextBlock();

        public function EventMirrorExample()
        {
            var myEvent:EventDispatcher = new EventDispatcher();

            myEvent.addEventListener("click", clickHandler);
            myEvent.addEventListener("mouseOut", mouseOutHandler);
            myEvent.addEventListener("mouseOver", mouseOverHandler);

            textElement2.eventMirror=myEvent;

            var groupVector:Vector.<ContentElement> = new Vector.<ContentElement>;
            groupVector.push(textElement1, textElement2, textElement3);
            var groupElement:GroupElement = new GroupElement(groupVector);

            textBlock.content = groupElement;
            createLines(textBlock);
        }

        private function clickHandler(event:MouseEvent):void
        {
            var newFont:FontDescription = new FontDescription();
            newFont.fontWeight = "bold";

            var newFormat:ElementFormat = new ElementFormat();
            newFormat.fontSize = 18;
            if(textElement3.text == "to see me in italic! ") {
                newFont.fontPosture = FontPosture.ITALIC;
                textElement3.replaceText(0,21, "to see me in normal font! ");
            }
            else {
                newFont.fontPosture = FontPosture.NORMAL;
                textElement3.replaceText(0, 26, "to see me in italic! ");
            }
            newFormat.fontDescription = newFont;
            textElement1.elementFormat = newFormat;
            textElement2.elementFormat = newFormat;
            textElement3.elementFormat = newFormat;
            createLines(textBlock);
        }
```

```
        private function mouseOverHandler(event:MouseEvent):void
        {
            Mouse.cursor = "button";
        }

        private function mouseOutHandler(event:MouseEvent):void
        {
                Mouse.cursor = "arrow";
        }

        private function createLines(textBlock:TextBlock):void
        {
            if(textBlock.firstLine)
                removeChild (textBlock.firstLine);
            var textLine:TextLine = textBlock.createTextLine (null, 300);
            textLine.x = 15;
            textLine.y = 20;
            addChild (textLine);
        }
    }
}
```

Die Funktionen `mouseOverHandler()` und `mouseOutHandler()` ändern den Cursor in einen Schaltflächen-Cursor, sobald sich dieser über dem Wort „here“ befindet, und zurück in einen Pfeil, wenn sich der Cursor über anderem Text befindet.

Formatieren von Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das `TextBlock`-Objekt ist eine Factory zum Erstellen von Textzeilen. Der Inhalt eines `TextBlock`-Objekts wird mithilfe des `TextElement`-Objekts zugewiesen. Das `ElementFormat`-Objekt legt die Formatierung des Textes fest. Die ElementFormat-Klasse definiert Eigenschaften wie die Ausrichtung der Grundlinie, Kerning, Laufweite, Textdrehung sowie Schriftgröße, Schriftfarbe und Groß-/Kleinschreibung. Er enthält außerdem ein `FontDescription`, welches detailliert unter „Arbeiten mit Schriftarten“ auf Seite 438 beschrieben wird.

Verwenden des ElementFormat-Objekts

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Der Konstruktor für das `ElementFormat`-Objekt lässt eine große Anzahl optionaler Parameter zu, unter anderem eine `FontDescription`. Sie können diese Eigenschaften auch außerhalb des Konstruktors festlegen. Im folgenden Beispiel wird die Beziehung zwischen den verschiedenen Objekten beim Definieren und Anzeigen einer einfachen Textzeile dargestellt:

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class ElementFormatExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef:ElementFormat;
        private var fd:FontDescription = new FontDescription();
        private var str:String;
        private var tl:TextLine;

        public function ElementFormatExample()
        {
            fd.fontName = "Garamond";
            ef = new ElementFormat(fd);
            ef.fontSize = 30;
            ef.color = 0xFF0000;
            str = "This is flash text";
            te = new TextElement(str, ef);
            tb.content = te;
            tl = tb.createTextLine(null,600);
            addChild(tl);
        }
    }
}
```

Schriftfarbe und Transparenz (Alpha)

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Eigenschaft `color` des `ElementFormat`-Objekts legt die Schriftfarbe fest. Der Wert ist eine Ganzzahl, die die RGB-Komponenten der Farbe angibt, z. B. 0xFF0000 für Rot und 0x00FF00 für Grün. Der Standardwert ist Schwarz (0x000000).

Die Eigenschaft `alpha` legt den Wert für die Alphatransparenz eines Elements (`TextElement` und `GraphicElement`) fest. Die möglichen Werte reichen von 0 (vollständig transparent) bis 1 (vollständig deckend, Standardwert). Elemente mit einem `alpha`-Wert von 0 sind zwar nicht sichtbar, jedoch trotzdem aktiv. Dieser Wert wird mit beliebigen übernommenen Alphawerten multipliziert, wodurch das Element transparenter dargestellt wird.

```
var ef:ElementFormat = new ElementFormat();
ef.alpha = 0.8;
ef.color = 0x999999;
```

Grundlinienausrichtung und -verschiebung

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die dominierende Grundlinie wird durch die Schriftart und -größe des größten Textes in einer Zeile bestimmt. Sie können diese Werte überschreiben, indem Sie `TextBlock.baselineFontDescription` und `TextBlock.baselineFontSize` festlegen. Die dominierende Grundlinie kann an einer der anderen Grundlinien innerhalb des Textes ausgerichtet werden. Zu diesen Grundlinien zählen die Oberlängen- und die Unterlängenlinie sowie die Oberseite, die Mitte oder die Unterseite von Ideogrammen.

***A.** Oberlänge **B.** Grundlinie **C.** Unterlänge **D.** x-Höhe*

Im Objekt `ElementFormat` werden die Merkmale der Grundlinie und Ausrichtung durch drei Eigenschaften bestimmt. Die Eigenschaft `alignmentBaseline` legt die Hauptgrundlinie eines `TextElement` oder `GraphicElement` fest. Diese Grundlinie ist die „Ausrichtungslinie“ für das Element, d. h. an dieser Position wird die dominierende Grundlinie des gesamten Textes ausgerichtet.

Die Eigenschaft `dominantBaseline` gibt an, welche der Grundlinien des Elements, durch die die vertikale Position des Elements auf der Linie bestimmt wird, zu verwenden ist. Der Standardwert ist `TextBaseline.ROMAN`, es kann jedoch auch die `IDEOGRAPHIC_TOP`- oder `IDEOGRAPHIC_BOTTOM`-Grundlinie als dominierende Grundlinie festgelegt werden.

Die Eigenschaft `baselineShift` verschiebt die Grundlinie um eine festgelegte Anzahl von Pixeln auf der y-Achse. In normalem Text (ohne Drehung) wird die Grundlinie bei einem positiven Wert nach unten und bei einem negativen Wert nach oben verschoben.

Typografische Groß-/Kleinschreibung

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Eigenschaft `TypographicCase` eines `ElementFormat`-Objekts gibt die Groß-/Kleinschreibung von Text an, z. B. Großbuchstaben, Kleinbuchstaben oder Kapitälchen.

```
var ef_Upper:ElementFormat = new ElementFormat();
ef_Upper.typographicCase = TypographicCase.UPPERCASE;

var ef_SmallCaps:ElementFormat = new ElementFormat();
ef_SmallCaps.typographicCase = TypographicCase.SMALL_CAPS;
```

Drehen von Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können einen Textblock oder die Zeichen (Glyphen) innerhalb eines Textsegments in Schritten von 90 Grad drehen. Die TextRotation-Klasse definiert die folgenden Konstanten, um die Drehung von Textblöcken und Zeichen festzulegen:

Konstante	Wert	Beschreibung
AUTO	„auto“	Gibt eine Drehung um 90 Grad gegen den Uhrzeigersinn an. Sie wird zumeist bei vertikalem asiatischem Text verwendet, um nur die erforderlichen Zeichen zu drehen.
ROTATE_0	„rotate_0“	Gibt keine Drehung an.
ROTATE_180	„rotate_180“	Gibt eine Drehung um 180 Grad an.
ROTATE_270	„rotate_270“	Gibt eine Drehung um 270 Grad an.
ROTATE_90	„rotate_90“	Gibt eine Drehung um 90 Grad im Uhrzeigersinn an.

Legen Sie zum Drehen der Textzeilen in einem Textblock die Eigenschaft `TextBlock.lineRotation` fest, bevor Sie die `TextBlock.createTextLine()`-Methode aufrufen, um die Textzeile zu erstellen.

Um die Zeichen (Glyphen) innerhalb eines Textblocks oder -segments zu drehen, legen Sie für die Eigenschaft `ElementFormat.textRotation` die Gradzahl fest, um die die Zeichen gedreht werden sollen. Als Glyphe wird die Form eines Zeichens oder eines Teils des Zeichens, das aus mehreren Glyphen besteht, bezeichnet. Der Buchstabe „a“ und der Punkt auf dem „i“ sind beispielsweise Glyphen.

Das Drehen von Glyphen ist vor allem in asiatischen Sprachen wichtig, um die Zeilen in eine vertikale Ausrichtung zu bringen, ohne die Zeichen innerhalb der Zeilen zu drehen. Weitere Informationen zum Drehen von asiatischem Text finden Sie unter „Ausrichten von asiatischem Text“ auf Seite 442.

Im folgenden Beispiel wird das Drehen eines Textblocks und der darin enthaltenen Zeichen veranschaulicht, wie z. B. bei asiatischem Text. Zudem wird in diesem Beispiel eine japanische Schriftart verwendet:

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class RotationExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef:ElementFormat;
        private var fd:FontDescription = new FontDescription();
        private var str:String;
        private var tl:TextLine;

        public function RotationExample()
        {
            fd.fontName = "MS Mincho";
            ef = new ElementFormat(fd);
            ef.textRotation = TextRotation.AUTO;
            str = "This is rotated Japanese text";
            te = new TextElement(str, ef);
            tb.lineRotation = TextRotation.ROTATE_90;
            tb.content = te;
            tl = tb.createTextLine(null,600);
            addChild(tl);
        }
    }
}
```

Sperren und Klonen des ElementFormat-Objekts

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wenn ein `ElementFormat`-Objekt einem beliebigen `ContentElement`-Typ zugewiesen ist, wird die Eigenschaft `locked` des Objekts automatisch auf `true` festgelegt. Wenn Sie versuchen, ein gesperrtes `ElementFormat`-Objekt zu bearbeiten, wird eine `IllegalOperationError`-Ausnahme ausgelöst. Die beste Vorgehensweise ist, ein solches Objekt vollständig zu definieren, bevor Sie es einer `TextElement`-Instanz zuweisen.

Wenn Sie eine vorhandene `ElementFormat`-Instanz ändern möchten, müssen Sie zunächst deren `locked`-Eigenschaft überprüfen. Wenn der Wert `true` lautet, können Sie mit der `clone()`-Methode eine ungesperrte Kopie des Objekts erstellen. Die Eigenschaften dieses ungesperrten Objekts können geändert werden, und anschließend kann des Objekt mit der `TextElement`-Instanz verknüpft werden. Alle aus dem Objekt neu erstellten Zeilen weisen die neue Formatierung auf. Zuvor von diesem Objekt erstellte Zeilen, die das alte Format verwenden, bleiben unverändert.

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class ElementFormatCloneExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var ef2:ElementFormat;
        private var fd:FontDescription = new FontDescription();

        public function ElementFormatCloneExample()
        {
            fd.fontName = "Garamond";
            ef1 = new ElementFormat(fd);
            ef1.fontSize = 24;
            var str:String = "This is flash text";
            te = new TextElement(str, ef);
            tb.content = te;
            var tx1:TextLine = tb.createTextLine(null,600);
            addChild(tx1);

            ef2 = (ef1.locked) ? ef1.clone() : ef1;
            ef2.fontSize = 32;
            tb.content.elementFormat = ef2;
            var tx2:TextLine = tb.createTextLine(null,600);
            addChild(tx2);
        }
    }
}
```

Arbeiten mit Schriftarten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das Objekt `FontDescription` wird zusammen mit dem Objekt `ElementFormat` verwendet, um die Schrift anzugeben und bestimmte Merkmale zu definieren. Hierzu zählen der Schriftname, die Schriftstärke, die Schriftneigung und die Schriftdarstellung sowie Angaben dazu, wie die Schriftart gefunden werden kann (Geräteschriftart oder eingebettete Schriftart).

Hinweis: *FTE unterstützt keine Type 1- oder Bitmap-Schriftarten wie Type 3, ATC, sfnt-wrapped CID oder Naked CID.*

Definieren von Schriftmerkmalen (FontDescription-Objekt)

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Eigenschaft `fontName` des `FontDescription`-Objekts kann einen einzelnen Namen oder eine durch Kommata getrennte Liste mit Namen angeben. Beispiel: In einer Liste wie „Arial, Helvetica, _sans“ sucht die Text Engine zunächst nach „Arial“, dann nach „Helvetica“ und zum Schluss nach „_sans“, wenn die beiden zuerst genannten Schriftarten nicht gefunden wurden. Die Schriftnamen umfassen drei allgemeine Geräteschriftarten: „_sans“, „_serif“ und „_typewriter“. Sie sind abhängig vom Wiedergabesystem spezifischen Geräteschriftarten zugewiesen. Es empfiehlt sich, in allen Schriftartbeschreibung, die Geräteschriftarten verwenden, Standardnamen ähnlich wie diese festzulegen. Wenn kein `fontName` angegeben ist, wird als Standardeinstellung „_serif“ verwendet.

Für die Eigenschaft `fontPosture` kann entweder der Standardwert (`FontPosture.NORMAL`) oder kursiv (`FontPosture.ITALIC`) festgelegt werden. Für die Eigenschaft `fontWeight` kann entweder der Standardwert (`FontWeight.NORMAL`) oder fett (`FontWeight.BOLD`) festgelegt werden.

```
var fd1:FontDescription = new FontDescription();
fd1.fontName = "Arial, Helvetica, _sans";
fd1.fontPosture = FontPosture.NORMAL;
fd1.fontWeight = FontWeight.BOLD;
```

Eingebettete Schriftarten und Geräteschriftarten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Eigenschaft `fontLookup` des `FontDescription`-Objekts gibt an, ob die Text Engine für die Textdarstellung nach einer Geräteschriftart oder nach einer eingebetteten Schriftart sucht. Wenn eine Geräteschriftart (`FontLookup.DEVICE`) angegeben wurde, sucht die Laufzeit auf dem Wiedergabesystem nach der Schriftart. Bei Angabe einer eingebetteten Schriftart (`FontLookup.EMBEDDED_CFF`) sucht die Laufzeit in der SWF-Datei nach einer eingebetteten Schriftart mit dem angegebenen Namen. Bei dieser Einstellung können nur eingebettete CFF-Schriftarten (Compact Font Format) verwendet werden. Wenn die angegebene Schriftart nicht gefunden wird, wird eine Fallback-Geräteschriftart verwendet.

Geräteschriftarten führen zu kleineren SWF-Dateigrößen. Mit eingebetteten Schriftarten bleibt das Erscheinungsbild über Plattformen hinweg gleich.

```
var fd1:FontDescription = new FontDescription();
fd1.fontLookup = FontLookup.EMBEDDED_CFF;
fd1.fontName = "Garamond, _serif";
```

Rendermodus und Hinting

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das CFF-Rendering (Compact Font Format) ist ab Flash Player 10 und Adobe AIR 1.5 verfügbar. Bei dieser Art der Schriftwiedergabe ist der Text besser lesbar und kleine Schriften werden mit besserer Qualität angezeigt. Diese Einstellung gilt nur für eingebettete Schriftarten. Das Objekt `FontDescription` verwendet `RenderingMode.CFF` als Standardeinstellung für die Eigenschaft `renderingMode`. Sie können für diese Eigenschaft auch `RenderingMode.NORMAL` festlegen, um die i die in Flash Player 7 oder früheren Versionen verwendete Darstellungsart zu nutzen.

Bei Auswahl von CFF-Rendering steuert eine zweite Eigenschaft, `cffHinting`, wie die horizontalen Ausläufer einer Schriftart in das Subpixel-Raster eingepasst werden. Der Standardwert `CFFHinting.HORIZONTAL_STEM` verwendet CFF-Hinting. Wenn Sie für diese Eigenschaft `CFFHinting.NONE` festlegen, wird das Hinting deaktiviert. Diese Einstellung eignet sich für Animationen oder große Schriftarten.

```
var fd1:FontDescription = new FontDescription();
fd1.renderingMode = RenderingMode.CFF;
fd1.cffHinting = CFFHinting.HORIZONTAL_STEM;
```

Sperren und Klonen des FontDescription-Objekts

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wenn ein `FontDescription`-Objekt einem `ElementFormat`-Objekt zugewiesen ist, wird die Eigenschaft `locked` des Objekts automatisch auf `true` festgelegt. Wenn Sie versuchen, ein gesperrtes `FontDescription`-Objekt zu bearbeiten, wird eine `IllegalOperationError`-Ausnahme ausgelöst. Die beste Vorgehensweise ist, ein solches Objekt vollständig zu definieren, bevor Sie es einer `ElementFormat`-Instanz zuweisen.

Wenn Sie ein vorhandenes `FontDescription`-Objekt ändern möchten, müssen Sie zunächst dessen `locked`-Eigenschaft überprüfen. Wenn der Wert `true` lautet, können Sie mit der `clone()`-Methode eine ungesperrte Kopie des Objekts erstellen. Die Eigenschaften dieses ungesperrten Objekts können geändert werden, und anschließend kann des Objekt mit der `ElementFormat`-Instanz verknüpft werden. Alle aus diesem `TextElement`-Objekt neu erstellten Zeilen weisen die neue Formatierung auf. Ältere Zeilen, die aus dem gleichen Objekt erstellt wurden, bleiben unverändert.

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class FontDescriptionCloneExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var ef2:ElementFormat;
        private var fd1:FontDescription = new FontDescription();
        private var fd2:FontDescription;

        public function FontDescriptionCloneExample()
        {
            fd1.fontName = "Garamond";
            ef1 = new ElementFormat(fd);
            var str:String = "This is flash text";
            te = new TextElement(str, ef);
            tb.content = te;
            var tx1:TextLine = tb.createTextLine(null,600);
            addChild(tx1);

            fd2 = (fd1.locked) ? fd1.clone() : fd1;
            fd2.fontName = "Arial";
            ef2 = (ef1.locked) ? ef1.clone() : ef1;
            ef2.fontDescription = fd2;
            tb.content.elementFormat = ef2;
            var tx2:TextLine = tb.createTextLine(null,600);
            addChild(tx2);
        }
    }
}
```

Steuern von Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

FTE bietet eine Reihe neuer Steuerungen zum Formatieren von Text, mit denen Sie die Ausrichtung und den Zeichenabstand (Kerning und Laufweite) bestimmen können. Weiterhin stehen Eigenschaften zur Verfügung, mit denen Sie Zeilenumbrüche steuern und Tabulatoren innerhalb von Zeilen festlegen können.

Ausrichten von Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Durch das Ausrichten des Textes als Blocksatz haben alle Zeilen in einem Absatz die gleiche Länge. Hierzu wird der Abstand zwischen Wörtern und teilweise auch zwischen den Buchstaben angepasst. Der Text wird hierdurch an beiden Seiten bündig ausgerichtet, während der Abstand zwischen den Wörtern und Buchstaben variiert. Textspalten in Zeitungen und Zeitschriften werden häufig auf diese Weise ausgerichtet.

Die Eigenschaft `lineJustfication` in der SpaceJustifier-Klasse ermöglicht Ihnen, die Ausrichtung von Zeilen in einem Textblock zu steuern. Die LineJustification-Klasse definiert Konstanten, mit deren Hilfe Sie eine Ausrichtungsoption angeben können: `ALL_BUT_LAST` zum Ausrichten des gesamten Textes mit Ausnahme der letzten Zeile; `ALL_INCLUDING_LAST` zum Ausrichten des gesamten Textes, einschließlich der letzten Zeile; `UNJUSTIFIED` (Standardeinstellung), um keine Ausrichtung für den Text festzulegen.

Legen Sie zum Ausrichten von Text die `lineJustification`-Eigenschaft auf eine Instanz der SpaceJustifier-Klasse und weisen Sie der Eigenschaft `textJustifier` einer TextBlock-Instanz diese Instanz zu. Im folgenden Beispiel wird ein Absatz erstellt, in dem der gesamte Text mit Ausnahme der letzten Textzeile ausgerichtet ist.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class JustifyExample extends Sprite
    {
        public function JustifyExample()
        {
            var str:String = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, " +
            "sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut " +
            "enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut " +
            "aliquip ex ea commodo consequat.";

            var format:ElementFormat = new ElementFormat();
            var textElement:TextElement=new TextElement(str,format);
            var spaceJustifier:SpaceJustifier=new
SpaceJustifier("en",LineJustification.ALL_BUT_LAST);

            var textBlock:TextBlock = new TextBlock();
            textBlock.content=textElement;
            textBlock.textJustifier=spaceJustifier;
            createLines(textBlock);
        }

        private function createLines(textBlock:TextBlock):void {
            var yPos=20;
            var textLine:TextLine=textBlock.createTextLine(null,150);

            while (textLine) {
                addChild(textLine);
                textLine.x=15;
                yPos+=textLine.textHeight+2;
                textLine.y=yPos;
                textLine=textBlock.createTextLine(textLine,150);
            }
        }
    }
}
```

Um den Abstand nicht nur zwischen Wörtern, sondern auch zwischen einzelnen Buchstaben zu variieren, legen Sie die Eigenschaft `SpaceJustifier.letterspacing` auf `true` fest. Durch das Aktivieren variabler Zeichenabstände werden unschöne Lücken zwischen einzelnen Wörtern vermieden, die bei einer einfachen Ausrichtung teilweise auftreten können.

Ausrichten von asiatischem Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Beim Ausrichten von asiatischem Text müssen weitere Aspekte berücksichtigt werden. Der Text wird von oben nach unten geschrieben, und einige Zeichen (so genannte Kinsoku) dürfen nicht am Anfang oder Ende einer Zeile stehen. Die JustificationStyle-Klasse definiert die folgenden Konstanten, die die Optionen zur Verarbeitung solcher Zeichen festlegen. Bei `PRIORITIZE_LEAST_ADJUSTMENT` basiert die Ausrichtung auf der Erweiterung oder Komprimierung der Zeile, abhängig davon, welche Methode das beste Ergebnis liefert. Bei `PUSH_IN_KINSOKU` basiert die Ausrichtung auf der Komprimierung von Kinsoku am Ende der Zeile oder auf dem Erweitern, falls kein Kinsoku vorhanden ist oder falls der Platz nicht ausreichend ist.

Bei `PUSH_OUT_ONLY` basiert die Ausrichtung auf einer Erweiterung der Zeile. Legen Sie zum Erstellen eines Blocks mit vertikalem asiatischem Text die `TextBlock.lineRotation`-Eigenschaft auf `TextRotation.ROTATE_90` und die `ElementFormat.textRotation`-Eigenschaft auf `TextRotation.AUTO` (Standardeinstellung) fest. Wenn Sie die Eigenschaft `textRotation` auf `AUTO` festlegen, bleiben die Zeichen in dem Text vertikal ausgerichtet und werden nicht geändert, wenn Sie die Zeile drehen. Bei der Einstellung `AUTO` erfolgt eine Drehung um 90 Grad gegen den Uhrzeigersinn für Zeichen mit voller Breite und breite Zeichen, wie von den Unicode-Eigenschaften des Zeichens festgelegt. Im folgenden Beispiel ist ein vertikaler Block mit japanischem Text zu sehen, der mit der `PUSH_IN_KINSOKU`-Option ausgerichtet wurde.

```
package
{
    import flash.text.engine.*;
    import flash.display.Stage;
    import flash.display.Sprite;
    import flash.system.Capabilities;

    public class EastAsianJustifyExample extends Sprite
    {
        public function EastAsianJustifyExample()
        {
            var Japanese_txt:String = String.fromCharCode(
            0x5185, 0x95A3, 0x5E9C, 0x304C, 0x300C, 0x653F, 0x5E9C, 0x30A4,
            0x30F3, 0x30BF, 0x30FC, 0x30CD, 0x30C3, 0x30C8, 0x30C6, 0x30EC,
            0x30D3, 0x300D, 0x306E, 0x52D5, 0x753B, 0x914D, 0x4FE1, 0x5411,
            0x3051, 0x306B, 0x30A2, 0x30C9, 0x30D3, 0x30B7, 0x30B9, 0x30C6,
            0x30E0, 0x30BA, 0x793E, 0x306E)
            var textBlock:TextBlock = new TextBlock();
            var font:FontDescription = new FontDescription();
            var format:ElementFormat = new ElementFormat();
            format.fontSize = 12;
            format.color = 0xCC0000;
            format.textRotation = TextRotation.AUTO;
            textBlock.baselineZero = TextBaseline.IDEOGRAPHIC_CENTER;
            var eastAsianJustifier:EastAsianJustifier = new EastAsianJustifier("ja",
LineJustification.ALL_BUT_LAST);
            eastAsianJustifier.justificationStyle = JustificationStyle.PUSH_IN_KINSOKU;
            textBlock.textJustifier = eastAsianJustifier;
            textBlock.lineRotation = TextRotation.ROTATE_90;
            var linePosition:Number = this.stage.stageWidth - 75;
            if (Capabilities.os.search("Mac OS") > -1)
                // set fontName: Kozuka Mincho Pro R
```

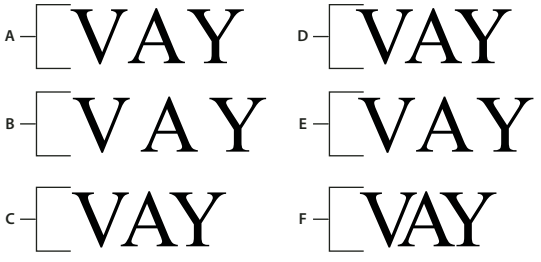
```
                font.fontName = String.fromCharCode(0x5C0F, 0x585A, 0x660E, 0x671D) + " Pro R";
            else
                font.fontName = "Kozuka Mincho Pro R";
            textBlock.content = new TextElement(Japanese_txt, format);
            var previousLine:TextLine = null;

            while (true)
            {
                var textLine:TextLine = textBlock.createTextLine(previousLine, 200);
                if (textLine == null)
                    break;
                textLine.y = 20;
                textLine.x = linePosition;
                linePosition -= 25;
                addChild(textLine);
                previousLine = textLine;
            }
        }
    }
}
```

Kerning und Laufweite

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Kerning und Laufweite wirken sich auf den Abstand zwischen nebeneinander liegenden Zeichenpaaren in einem Textblock aus. Das Kerning steuert die Ausrichtung von Zeichenpaaren aneinander, z. B. bei den Paaren „WA“ oder „Va“. Kerning wird im `ElementFormat`-Objekt gesetzt. Standardmäßig ist das Kerning aktiviert (`Kerning.ON`), Sie können jedoch auch OFF oder AUTO einstellen. In letzterem Fall wird das Kerning nur zwischen Zeichen angewendet, wenn dies nicht Kanji-, Hiragana- oder Katakana-Zeichen sind.

Laufweite bedeutet, dass zwischen allen Zeichen in einem Textblock eine bestimmte Anzahl von Pixeln hinzugefügt oder entfernt wird. Die Laufweite wird ebenfalls im Objekt `ElementFormat` eingestellt. Sie kann sowohl in eingebetteten Schriftarten als auch in Geräteschriftarten verwendet werden. FTE unterstützt zwei Eigenschaften für die Laufweite: `trackingLeft` zum Hinzufügen/Entfernen von Pixeln auf der linken Seite eines Zeichens und `trackingRight` zum Hinzufügen/Entfernen von Pixeln auf der rechten Seite. Bei Verwendung von Kerning wird der Laufweitenwert für jedes Zeichenpaar zu den Kerning-Werten addiert bzw. von diesen subtrahiert.



***A.** Kerning.OFF **B.** TrackingRight=5, Kerning.OFF **C.** TrackingRight=-5, Kerning.OFF **D.** Kerning.ON **E.** TrackingRight=-5, Kerning.ON **F.** TrackingRight=-5, Kerning.ON*

```
var ef1:ElementFormat = new ElementFormat();
ef1.kerning = Kerning.OFF;

var ef2:ElementFormat = new ElementFormat();
ef2.kerning = Kerning.ON;
ef2.trackingLeft = 0.8;
ef2.trackingRight = 0.8;

var ef3:ElementFormat = new ElementFormat();
ef3.trackingRight = -0.2;
```

Zeilenumbrüche für umgebenden Text

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die Eigenschaft `breakOpportunity` des `ElementFormat`-Objekts legt fest, welche Zeichen für Zeilenumbrüche verwendet werden können, wenn der umgebende Text in mehrere Zeilen aufgeteilt ist. Bei der Standardeinstellung `BreakOpportunity.AUTO` werden standardmäßige Unicode-Eigenschaften verwendet, z. B. Umbruch zwischen Wörtern und bei Bindestrichen. Bei Verwendung von `BreakOpportunity.ALL` ist bei jedem Zeichen ein Zeilenumbruch möglich. Dies ist hilfreich, wenn Sie bestimmte Effekte erzeugen möchten, z. B. um Text entlang eines Pfads anzuordnen.

```
var ef:ElementFormat = new ElementFormat();
ef.breakOpportunity = BreakOpportunity.ALL;
```

Tabulatoren

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wenn Sie Tabulatoren in einem Textblock festlegen möchten, müssen Sie diese definieren, indem Sie Instanzen der TabStop-Klasse erstellen. Die Parameter für den `TabStop()`-Konstruktor geben an, wie der Text an den Tabulatoren ausgerichtet wird. Diese Parameter legen die Position der Tabulatoren und bei einer dezimalen Ausrichtung den Wert für die Ausrichtung (angegeben als Zeichenfolge) fest. Dieser Wert ist zumeist ein Dezimalpunkt, es kann jedoch beispielsweise auch ein Komma, ein Dollarzeichen oder das Symbol für Yen oder Euro verwendet werden. Durch die folgende Codezeile wird ein Tabulator mit dem Namen „tab1“ erstellt.

```
var tab1:TabStop = new TabStop(TabAlignment.DECIMAL, 50, ".");
```

Nachdem Sie die Tabulatoren für einen Textblock erstellt haben, müssen Sie diese zur Eigenschaft `tabStops` einer TextBlock-Instanz zuweisen. Da die Eigenschaft `tabStops` einen Vektor voraussetzt, müssen Sie jedoch zunächst einen Vektor erstellen und diesem die Tabulatoren hinzufügen. Der Vektor ermöglicht Ihnen, dem Textblock eine Gruppe von Tabulatoren zuzuweisen. Im folgenden Beispiel wird eine `Vector<TabStop>`-Instanz erstellt und eine Gruppe von TabStop-Objekten zu dieser Instanz hinzugefügt. Anschließend werden die Tabulatoren zur Eigenschaft `tabStops` einer TextBlock-Instanz zugeordnet.

```
var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
tabStops.push(tab1, tab2, tab3, tab4);
textBlock.tabStops = tabStops
```

Weitere Informationen zu Vectors finden Sie unter „Verwenden von Arrays“ auf Seite 25.

Im folgenden Beispiel werden die Auswirkungen der verschiedenen TabStop-Ausrichtungsoptionen veranschaulicht.

```
package {

    import flash.text.engine.*;
    import flash.display.Sprite;

    public class TabStopExample extends Sprite
    {
        public function TabStopExample()
        {
            var format:ElementFormat = new ElementFormat();
            format.fontDescription = new FontDescription("Arial");
            format.fontSize = 16;

            var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
            tabStops.push(
                new TabStop(TabAlignment.START, 20),
                new TabStop(TabAlignment.CENTER, 140),
                new TabStop(TabAlignment.DECIMAL, 260, "."),
                new TabStop(TabAlignment.END, 380));
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = new TextElement(
                "\tt1\tt2\tt3\tt4\n" +
                "\tThis line aligns on 1st tab\n" +
                "\t\t\t\tThis is the end\n" +
                "\tThe following fragment centers on the 2nd tab:\t\t\n" +
                "\t\tit's on me\t\t\n" +
                "\tThe following amounts align on the decimal point:\n" +
                "\t\t\t45.00\t\n" +
                "\t\t\t75,320.00\t\n" +
                "\t\t\t6,950.00\t\n" +
                "\t\t\t7.01\t\n", format);

            textBlock.tabStops = tabStops;
            var yPosition:Number = 60;
            var previousTextLine:TextLine = null;
            var textLine:TextLine;
            var i:int;
            for (i = 0; i < 10; i++) {
                textLine = textBlock.createTextLine(previousTextLine, 1000, 0);
                textLine.x = 20;
                textLine.y = yPosition;
                addChild(textLine);
                yPosition += 25;
                previousTextLine = textLine;
            }
        }
    }
}
```

Flash Text Engine-Beispiel: Zeichnungslayout

Flash Player 10 und höher, Adobe AIR 1.5 und höher

In diesem Programmierbeispiel wird gezeigt, wie die Flash Text Engine das Layout einer einfachen Zeitungsseite gestaltet. Die Seite enthält eine große Schlagzeile, eine Überschrift und einen aus mehreren Spalten bestehenden Text.

Erstellen Sie zunächst eine FLA-Datei und verknüpfen Sie den folgenden Code mit Frame 2 der Standardebene:

```
import com.example.programmingas3.newslayout.StoryLayout ;
// frame sc ript - create  a 3-columned arti cle layout
var story:StoryLayout = new StoryLayout(720, 500, 3, 10);
story.x = 20;
story.y = 80;
addChild(story);
stop();
```

Das Steuerungsskript für dieses Beispiel ist StoryLayout.as. Es setzt den Inhalt, liest die Stilinformationen von einem externen Stylesheet ein und weist diese Stile den ElementFormat-Objekten zu. Es erstellt dann die Schlagzeile, die Überschrift und den mehrspaltigen Textkörper.

```
package com.example.programmingas3.newslayout
{
    import flash.display.Sprite;
    import flash.text.StyleSheet;
    import flash.text.engine.*;

    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.net.URLLoader;
    import flash.display.Sprite;
    import flash.display.Graphics;

    public class StoryLayout extends Sprite
    {
        public var headlineTxt:HeadlineTextField;
        public var subtitleTxt:HeadlineTextField;
        public var storyTxt:MultiColumnText;
        public var sheet:StyleSheet;
        public var h1_ElFormat:ElementFormat;
        public var h2_ElFormat:ElementFormat;
        public var p_ElFormat:ElementFormat;

        private var loader:URLLoader;

        public var paddingLeft:Number;
        public var paddingRight:Number;
        public var paddingTop:Number;
        public var paddingBottom:Number;

        public var preferredWidth:Number;
        public var preferredHeight:Number;

        public var numColumns:int;

        public var bgColor:Number = 0xFFFFFF;
```

```
        public var headline:String = "News Layout Example";
        public var subtitle:String = "This example formats text like a newspaper page using the
Flash Text Engine API. ";

        public var rawTestData:String =
        "From the part Mr. Burke took in the American Revolution, it was natural that I should
consider him a friend to mankind; and as our acquaintance commenced on that ground, it would
have been more agreeable to me to have had cause to continue in that opinion than to change it. " +
        "At the time Mr. Burke made his violent speech last winter in the English Parliament
against the French Revolution and the National Assembly, I was in Paris, and had written to him
but a short time before to inform him how prosperously matters were going on. Soon after this I
saw his advertisement of the Pamphlet he intended to publish: As the attack was to be made in a
language but little studied, and less understood in France, and as everything suffers by
translation, I promised some of the friends of the Revolution in that country that whenever Mr.
Burke's Pamphlet came forth, I would answer it. This appeared to me the more necessary to be
done, when I saw the flagrant misrepresentations which Mr. Burke's Pamphlet contains; and that
while it is an outrageous abuse on the French Revolution, and the principles of Liberty, it is
an imposition on the rest of the world. " +
        "I am the more astonished and disappointed at this conduct in Mr. Burke, as (from the
circumstances I am going to mention) I had formed other expectations. " +
        "I had seen enough of the miseries of war, to wish it might never more have existence
in the world, and that some other mode might be found out to settle the differences that should
occasionally arise in the neighbourhood of nations. This certainly might be done if Courts were
disposed to set honesty about it, or if countries were enlightened enough not to be made the
dupes of Courts. The people of America had been bred up in the same prejudices against France,
which at that time characterised the people of England; but experience and an acquaintance with
the French Nation have most effectually shown to the Americans the falsehood of those prejudices;
and I do not believe that a more cordial and confidential intercourse exists between any two
countries than between America and France. ";

        public function StoryLayout(w:int = 400, h:int = 200, cols:int = 3, padding:int =
10):void
        {
            this.preferredWidth = w;
            this.preferredHeight = h;

            this.numColumns = cols;

            this.paddingLeft = padding;
            this.paddingRight = padding;
            this.paddingTop = padding;
            this.paddingBottom = padding;

            var req:URLRequest = new URLRequest("story.css");
            loader = new URLLoader();
            loader.addEventListener(Event.COMPLETE, onCSSFileLoaded);
            loader.load(req);
        }

        public function onCSSFileLoaded(event:Event):void
        {
            this.sheet = new StyleSheet();
            this.sheet.parseCSS(loader.data);

            // convert headline styles to ElementFormat objects
            h1_ElFormat = getElFormat("h1", this.sheet);
```

```
        h1_ElFormat.typographicCase = TypographicCase.UPPERCASE;
        h2_ElFormat = getElFormat("h2", this.sheet);
        p_ElFormat = getElFormat("p", this.sheet);
        displayText();
    }

    public function drawBackground():void
    {
        var h:Number = this.storyTxt.y + this.storyTxt.height +
                        this.paddingTop + this.paddingBottom;
        var g:Graphics = this.graphics;
        g.beginFill(this.bgColor);
        g.drawRect(0, 0, this.width + this.paddingRight + this.paddingLeft, h);
        g.endFill();
    }

    /**
     * Reads a set of style properties for a named style and then creates
     * a TextFormat object that uses the same properties.
     */
    public function getElFormat(styleName:String, ss:StyleSheet):ElementFormat
    {
    var style:Object = ss.getStyle(styleName);
    if (style != null)
    {
    var colorStr:String = style.color;
    if (colorStr != null && colorStr.indexOf("#") == 0)
    {
        style.color = colorStr.substr(1);
    }
            var fd:FontDescription = new FontDescription(
                                    style.fontFamily,
                                    style.fontWeight,
                                    FontPosture.NORMAL,
                                    FontLookup.DEVICE,
                                    RenderingMode.NORMAL,
                                    CFFHinting.NONE);
    var format:ElementFormat = new ElementFormat(fd,
                            style.fontSize,
                            style.color,
                            1,
                            TextRotation.AUTO,
                            TextBaseline.ROMAN,
                            TextBaseline.USE_DOMINANT_BASELINE,
                            0.0,
                            Kerning.ON,
                            0.0,
                            0.0,
                            "en",
                                    BreakOpportunity.AUTO,
                                    DigitCase.DEFAULT,
                                    DigitWidth.DEFAULT,
                            LigatureLevel.NONE,
                                    TypographicCase.DEFAULT);

    if (style.hasOwnProperty("letterSpacing"))
    {
```

```
            format.trackingRight = style.letterSpacing;
        }
        }
        return format;
        }

        public function displayText():void
        {
        headlineTxt = new HeadlineTextField(h1_ElFormat,headline,this.preferredWidth);
        headlineTxt.x = this.paddingLeft;
        headlineTxt.y = 40 + this.paddingTop;
            headlineTxt.fitText(1);
            this.addChild(headlineTxt);

        subtitleTxt = new HeadlineTextField(h2_ElFormat,subtitle,this.preferredWidth);
        subtitleTxt.x = this.paddingLeft;
        subtitleTxt.y = headlineTxt.y + headlineTxt.height;
            subtitleTxt.fitText(2);
        this.addChild(subtitleTxt);

        storyTxt = new MultiColumnText(rawTestData, this.numColumns,
                        20, this.preferredWidth, this.preferredHeight, p_ElFormat);
        storyTxt.x = this.paddingLeft;
        storyTxt.y = subtitleTxt.y + subtitleTxt.height + 10;
        this.addChild(storyTxt);

            drawBackground();
        }
    }
}
```

FormattedTextBlock.as wird als Basisklasse für die Erstellung von Textblöcken verwendet. Es enthält außerdem Dienstfunktionen für die Änderung der Schriftgröße und der Groß- und Kleinschreibung.

```
package com.example.programmingas3.newslayout
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class FormattedTextBlock extends Sprite
    {
        public var tb:TextBlock;
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var textWidth:int;
        public var totalTextLines:int;
        public var blockText:String;
        public var leading:Number = 1.25;
        public var preferredWidth:Number = 720;
        public var preferredHeight:Number = 100;

        public function FormattedTextBlock(ef:ElementFormat,txt:String, colW:int = 0)
        {
            this.textWidth = (colW==0) ? preferredWidth : colW;
            blockText = txt;
            ef1 = ef;
            tb = new TextBlock();
```

```
        tb.textJustifier = new SpaceJustifier("en",LineJustification.UNJUSTIFIED,false);
        te = new TextElement(blockText,this.ef1);
        tb.content = te;
        this.breakLines();
    }

    private function breakLines()
    {
        var textLine:TextLine = null;
        var y:Number = 0;
        var lineNum:int = 0;
        while (textLine = tb.createTextLine(textLine,this.textWidth,0,true))
        {
            textLine.x = 0;
            textLine.y = y;
            y += this.leading*textLine.height;
            this.addChild(textLine);
        }
        for (var i:int = 0; i < this.numChildren; i++)
        {
            TextLine(this.getChildAt(i)).validity = TextLineValidity.STATIC;
        }
        this.totalTextLines = this.numChildren;
    }

    private function rebreakLines()
    {
        this.clearLines();
        this.breakLines();
    }

    private function clearLines()
    {
        while(this.numChildren)
        {
            this.removeChildAt(0);
        }
    }
```

```
        public function changeSize(size:uint=12):void
        {
            if (size > 5)
            {
                var ef2:ElementFormat = ef1.clone();
                ef2.fontSize = size;
                te.elementFormat = ef2;
                this.rebreakLines();
            }
        }

        public function changeCase(newCase:String = "default"):void
        {
            var ef2:ElementFormat = ef1.clone();
            ef2.typographicCase = newCase;
            te.elementFormat = ef2;
        }
    }
}
```

HeadlineTextBlock.as erweitert die FormattedTextBlock-Klasse und wird zur Erstellung von Schlagzeilen verwendet. Es enthält eine Funktion zur Einpassung von Text in einen vorgeschriebenen Rahmen auf der Seite.

```
package com.example.programmingas3.newslayout
{
    import flash.text.engine.*;
    public class HeadlineTextField extends FormattedTextBlock
    {

        public static var MIN_POINT_SIZE:uint = 6;
        public static var MAX_POINT_SIZE:uint = 128;

        public function HeadlineTextField(te:ElementFormat,txt:String,colW:int = 0)
        {
            super(te,txt);
        }

        public function fitText(maxLines:uint = 1, targetWidth:Number = -1):uint
        {
            if (targetWidth == -1)
            {
                targetWidth = this.width;
            }

            var pixelsPerChar:Number = targetWidth / this.blockText.length;
            var pointSize:Number = Math.min(MAX_POINT_SIZE,
                            Math.round(pixelsPerChar * 1.8 * maxLines));

            if (pointSize < 6)
            {
                // the point size is too small
                return pointSize;
            }

            this.changeSize(pointSize);
            if (this.totalTextLines > maxLines)
```

```
            {
                return shrinkText(--pointSize, maxLines);
            }
            else
            {
                return growText(pointSize, maxLines);
            }
        }

        public function growText(pointSize:Number, maxLines:uint = 1):Number
        {
            if (pointSize >= MAX_POINT_SIZE)
            {
                return pointSize;
            }

            this.changeSize(pointSize + 1);
            if (this.totalTextLines > maxLines)
            {
                // set it back to the last size
                this.changeSize(pointSize);
                return pointSize;
            }
            else
            {
                return growText(pointSize + 1, maxLines);
            }
        }

        public function shrinkText(pointSize:Number, maxLines:uint=1):Number
        {
            if (pointSize <= MIN_POINT_SIZE)
            {
                return pointSize;
            }
            this.changeSize(pointSize);

            if (this.totalTextLines > maxLines)
            {
                return shrinkText(pointSize - 1, maxLines);
            }
            else
            {
                return pointSize;
            }
        }
    }
}
```

MultiColumnText.as formatiert Text in mehrspaltigen Textkörpern. Es zeigt die flexible Verwendung eines TextBlock-Objekts, das zum Erstellen, Formatieren und Positionieren von Textzeilen dient.

```
package com.example.programmingas3.newslayout
{
    import flash.display.Sprite;
    import flash.text.engine.*;

    public class MultiColumnText extends Sprite
    {
        private var tb:TextBlock;
        private var te:TextElement;
        private var numColumns:uint = 2;
        private var gutter:uint = 10;
        private var leading:Number = 1.25;
        private var preferredWidth:Number = 400;
        private var preferredHeight:Number = 100;
        private var colWidth:int = 200;

        public function MultiColumnText(txt:String = "",cols:uint = 2,
            gutter:uint = 10, w:Number = 400, h:Number = 100,
            ef:ElementFormat = null):void
        {
            this.numColumns = Math.max(1, cols);
            this.gutter = Math.max(1, gutter);

            this.preferredWidth = w;
            this.preferredHeight = h;

            this.setColumnWidth();

            var field:FormattedTextBlock = new FormattedTextBlock(ef,txt,this.colWidth);
            var totLines:int = field.totalTextLines;
            field = null;
            var linesPerCol:int = Math.ceil(totLines/cols);

            tb = new TextBlock();
            te = new TextElement(txt,ef);
            tb.content = te;
            var textLine:TextLine = null;
            var x:Number = 0;
            var y:Number = 0;
            var i:int = 0;
            var j:int = 0;
            while (textLine = tb.createTextLine(textLine,this.colWidth,0,true))
            {
                textLine.x = Math.floor(i/(linesPerCol+1))*(this.colWidth+this.gutter);
                textLine.y = y;
                y += this.leading*textLine.height;
```

```
                j++;
                if(j>linesPerCol)
                {
                    y = 0;
                    j = 0;
                }
                i++;

                this.addChild(textLine);
            }
        }

        private function setColumnWidth():void
        {
        this.colWidth = Math.floor( (this.preferredWidth -
            ((this.numColumns - 1) * this.gutter)) / this.numColumns);
        }

    }
}
```

Kapitel 23: Verwenden des Text Layout Framework

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Übersicht über das Text Layout Framework

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das Text Layout Framework (TLF) ist eine erweiterbare ActionScript-Bibliothek. Das TLF basiert auf der Textengine in Adobe® Flash® Player 10 und Adobe® AIR® 1.5. Das TLF bietet erweiterte typografische Funktionen sowie Textlayout-Funktionen für innovative Typografie im Web. Das Framework kann mit Adobe® Flex® oder Adobe® Flash® Professional verwendet werden. Entwickler können vorhandene Komponenten verwenden oder erweitern oder mit dem Framework eigene Textkomponenten erstellen.

Das TLF umfasst die folgenden Funktionsmerkmale:

- Bidirektionaler Text, vertikaler Text und mehr als 30 Schriften, darunter Arabisch, Hebräisch, Chinesisch, Japanisch, Koreanisch, Thai, Laotisch und Vietnamesisch.
- Auswählen, Bearbeiten und Anordnen von Text über mehrere Spalten und verknüpfte Container.
- Vertikaler Text, Tate-Chu-Yoko (horizontaler Textblock innerhalb von vertikalen Textzeilen) und Ausrichtung für ostasiatische Typografie.
- Zahlreiche typografische Steuerelemente, beispielsweise für Kerning, Ligaturen, typografische Buchstabenart, Zifferndarstellung und Ziffernbreite sowie weiche Bindestriche.
- Ausschneiden, Kopieren, Einfügen, Rückgängigmachen und standardmäßige Tastatur- und Mausgesten zur Bearbeitung.
- Vielseitige Entwickler-APIs zum Bearbeiten von Textinhalt, Layout und Markup sowie zum Erstellen von benutzerdefinierten Textkomponenten.
- Robuste Unterstützung für Listen, wie benutzerdefinierte Markierungen und Nummerierungsformate.
- Inlinebilder und Positionierungsregeln.

Das TLF ist eine ActionScript 3.0-Bibliothek, die auf der neuen Flash Text Engine (FTE) basiert, die in Flash Player 10 eingeführt wurde. FTE kann über das `flash.text.engine`-Paket aufgerufen werden, das in der Application Programming Interface (API) von Flash Player 10 enthalten ist.

Die Flash Player-API bietet der Text-Engine jedoch nur Zugriff auf niedriger Ebene, d. h. dass für einige Aufgaben relativ viel Code notwendig ist. Das TLF fasst den Code auf niedriger Ebene in einfacheren APIs zusammen. Außerdem bietet das TLF eine konzeptionelle Architektur, die die von der FTE definierten grundlegenden Bausteine in einem System organisiert, das sich einfacher verwenden lässt.

Im Gegensatz zur FTE ist das TLF nicht in Flash Player integriert. Es handelt sich vielmehr um eine unabhängige Bibliothek, die vollständig in ActionScript 3.0 geschrieben ist. Da das Framework erweiterbar ist, lässt es sich an bestimmte Umgebungen anpassen. Sowohl Flash Professional als auch das Flex SDK enthalten Komponenten, die auf dem TLF-Framework basieren.

Verwandte Hilfethemen

„Fluss“-TLF-Markupanwendung

Unterstützung von komplexen Schriften

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das TLF unterstützt komplexe Schriften. Beispielsweise kann Text in Sprachen, die von rechts nach links geschrieben werden, angezeigt und bearbeitet werden. Das TLF unterstützt auch das Anzeigen und Bearbeiten von Schriften in gemischter Schreibweise (links nach rechts und rechts nach links), wie Arabisch und Hebräisch. Das Framework unterstützt nicht nur das vertikale Textlayout für Chinesisch, Japanisch und Koreanisch, sondern auch TCY-Elemente (Tate-Chuu-Yoko). TCY-Elemente sind Blöcke von horizontalem Text, die in eine vertikale Textzeile eingebettet sind. Nachstehende Schriften werden unterstützt:

- Latein (Englisch, Spanisch, Französisch, Vietnamesisch usw.)
- Griechisch, Kyrillisch, Armenisch, Georgisch und Äthiopisch
- Arabisch und Hebräisch
- Han-Ideogramme und Kana (Chinesisch, Japanisch und Koreanisch) sowie Hangul Johab (Koreanisch)
- Thai, Laotisch und Khmer
- Devanagari, Bengalisch, Gurmukhi, Malayalam, Telugu, Tamil, Gujarati, Oriya, Kannada und Tibetisch
- Tifinagh, Yi, Cherokee, kanadische Silbenzeichen, Deseret, Schawisch, Vai, Tagalog, Hanunoo, Buhid und Tagbanwa

Verwenden des Text Layout Framework in Flash Professional und Flex

Sie können die TLF-Klassen direkt zur Erstellung von benutzerdefinierten Komponenten in Flash verwenden. Außerdem bietet Flash Professional CS5 die neue fl.text.TLFTextField-Klasse, die die TLF-Funktionalität einschließt. Mit der TLFTextField-Klasse können Sie in ActionScript Textfelder erstellen, die die erweiterten TLF-Funktionen für die Textanzeige nutzen. Erstellen Sie ein TLFTextField-Objekt auf die gleiche Weise wie Sie ein Textfeld mit der TextField-Klasse erstellen. Verwenden Sie dann die `textFlow`-Eigenschaft, um erweiterte Formatierung aus den TLF-Klassen zuzuweisen.

In Flash Professional können Sie die TLFTextField-Instanz auch mithilfe des Textwerkzeugs auf der Bühne erstellen. Dann können Sie ActionScript verwenden, um Formatierung und Layout des Inhalts im Textfeld mithilfe der TLF-Klassen zu steuern. Weitere Informationen finden Sie im Abschnitt zu TLFTextField im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Wenn Sie in Flex arbeiten, verwenden Sie die TLF-Klassen. Weitere Informationen finden Sie unter „Verwenden des Text Layout Framework“ auf Seite 457.

Verwenden des Text Layout Framework

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Wenn Sie in Flex arbeiten oder benutzerdefinierte Textkomponenten erstellen, verwenden Sie die TLF-Klassen. Das TLF ist eine ActionScript 3.0-Bibliothek, die vollständig in der textLayout.swc-Bibliothek enthalten ist. Die TLF-Bibliothek enthält etwa 100 ActionScript 3.0-Klassen und Schnittstellen, die in 10 Pakete unterteilt sind. Diese Pakete sind Unterpakete des flashx.textLayout-Pakets.

Die TLF-Klassen

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die TLF-Klassen lassen sich in drei Kategorien einteilen:

- Klassen für die Datenstrukturierung und Formatierung
- Klassen für das Rendern
- Klassen für die Benutzerinteraktion

Klassen für die Datenstrukturierung und Formatierung

Die folgenden Pakete enthalten die TLF-Klassen für die Datenstrukturierung und Formatierung:

- flashx.textLayout.elements
- flashx.textLayout.formats
- flashx.textLayout.conversion

Die Hauptdatenstruktur des TLF ist die Textflusshierarchie, die im elements-Paket definiert ist. Innerhalb der Struktur können Sie Textzeilen mithilfe des format-Pakets Stile und Attribute zuweisen. Mit dem conversion-Paket können Sie steuern, wie Text in die Datenstruktur importiert und aus der Datenstruktur exportiert wird.

Klassen für das Rendern

Die folgenden Pakete enthalten die TLF-Klassen für das Rendern:

- flashx.textLayout.factory
- flashx.textLayout.container
- flashx.textLayout.compose

Mit den Klassen in diesen Paketen kann Text zur Anzeige in Flash Player gerendert werden. Das factory-Paket bietet eine einfache Art, statischen Text anzuzeigen. Das container-Paket enthält Klassen und Schnittstellen, die Anzeigecontainer für dynamischen Text definieren. Das compose-Paket definiert Techniken für die Positionierung und Anzeige von dynamischen Texten in Containern.

Klassen für die Benutzerinteraktion

Die folgenden Pakete enthalten die TLF-Klassen für die Benutzerinteraktion:

- flashx.textLayout.edit
- flashx.textLayout.operations
- flashx.textLayout.events

Die edit- und operations-Pakete definieren Klassen für die Bearbeitung von Text, der in den Datenstrukturen gespeichert ist. Das events-Paket enthält Klassen für die Ereignisverarbeitung.

Allgemeine Schritte zum Erstellen von Text mit dem Text Layout Framework

In den folgenden Schritten wird das allgemeine Verfahren zum Erstellen von Text mit dem Text Layout Framework beschrieben:

1. Importieren Sie formatierten Text in die TLF-Datenstrukturen. Weitere Informationen finden Sie unter „Strukturieren von Text mit dem TLF“ auf Seite 462 und „Formatieren von Text mit dem TLF“ auf Seite 466.
2. Erstellen Sie mindestens einen verknüpften Anzeigeobjektcontainer für den Text. Weitere Informationen finden Sie unter „Verwalten von Textcontainern mit dem TLF“ auf Seite 468.
3. Ordnen Sie den Text in den Datenstrukturen den Containern zu und legen Sie Optionen für die Bearbeitung und für den Bildlauf fest. Weitere Informationen finden Sie unter „Aktivieren von Textauswahl, Bearbeitung und Rückgängigmachen mit dem TLF“ auf Seite 469.
4. Erstellen Sie eine Ereignisprozedur, mit der der Text nach einer Größenänderung (oder nach anderen Ereignissen) neu angeordnet wird. Weitere Informationen finden Sie unter „Ereignisverarbeitung mit dem TLF“ auf Seite 469.

Text Layout Framework-Beispiel: Zeitungslayout

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das folgende Beispiel veranschaulicht, wie mit dem TLF das Layout einer einfachen Zeitungsseite gestaltet werden kann. Die Seite enthält eine große Schlagzeile, eine Überschrift und einen aus mehreren Spalten bestehenden Text:

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.Event;
    import flash.geom.Rectangle;

    import flashx.textLayout.compose.StandardFlowComposer;
    import flashx.textLayout.container.ContainerController;
    import flashx.textLayout.container.ScrollPolicy;
    import flashx.textLayout.conversion.TextConverter;
    import flashx.textLayout.elements.TextFlow;
    import flashx.textLayout.formats.TextLayoutFormat;

    public class TLFNewsLayout extends Sprite
    {
        private var hTextFlow:TextFlow;
        private var headContainer:Sprite;
        private var headlineController:ContainerController;
        private var hContainerFormat:TextLayoutFormat;

        private var bTextFlow:TextFlow;
        private var bodyTextContainer:Sprite;
        private var bodyController:ContainerController;
        private var bodyTextContainerFormat:TextLayoutFormat;

        private const headlineMarkup:String = "<flow:TextFlow
```

```
xmlns:flow='http://ns.adobe.com/textLayout/2008'><flow:p textAlign='center'><flow:span
fontFamily='Helvetica' fontSize='18'>TLF News Layout Example</flow:span><flow:br/><flow:span
fontFamily='Helvetica' fontSize='14'>This example formats text like a newspaper page with a
headline, a subtitle, and multiple columns</flow:span></flow:p></flow:TextFlow>";

        private const bodyMarkup:String = "<flow:TextFlow
xmlns:flow='http://ns.adobe.com/textLayout/2008' fontSize='12' textIndent='10' marginBottom='15'
paddingTop='4' paddingLeft='4'><flow:p marginBottom='inherit'><flow:span>There are many
</flow:span><flow:span fontStyle='italic'>such</flow:span><flow:span> lime-kilns in that tract
of country, for the purpose of burning the white marble which composes a large part of the
substance of the hills. Some of them, built years ago, and long deserted, with weeds growing in
the vacant round of the interior, which is open to the sky, and grass and wild-flowers rooting
themselves into the chinks of the stones, look already like relics of antiquity, and may yet be
overspread with the lichens of centuries to come. Others, where the lime-burner still feeds his
daily and nightlong fire, afford points of interest to the wanderer among the hills, who seats
himself on a log of wood or a fragment of marble, to hold a chat with the solitary man. It is a
lonesome, and, when the character is inclined to thought, may be an intensely thoughtful
occupation; as it proved in the case of Ethan Brand, who had mused to such strange purpose, in
days gone by, while the fire in this very kiln was burning.</flow:span></flow:p><flow:p
marginBottom='inherit'><flow:span>The man who now watched the fire was of a different order, and
troubled himself with no thoughts save the very few that were requisite to his business. At
frequent intervals, he flung back the clashing weight of the iron door, and, turning his face
from the insufferable glare, thrust in huge logs of oak, or stirred the immense brands with a
long pole. Within the furnace were seen the curling and riotous flames, and the burning marble,
almost molten with the intensity of heat; while without, the reflection of the fire quivered on
the dark intricacy of the surrounding forest, and showed in the foreground a bright and ruddy
little picture of the hut, the spring beside its door, the athletic and coal-begrimed figure of
the lime-burner, and the half-frightened child, shrinking into the protection of his father's
shadow. And when again the iron door was closed, then reappeared the tender light of the half-
full moon, which vainly strove to trace out the indistinct shapes of the neighboring mountains;
and, in the upper sky, there was a flitting congregation of clouds, still faintly tinged with the
rosy sunset, though thus far down into the valley the sunshine had vanished long and long
ago.</flow:span></flow:p></flow:TextFlow>";

        public function TLFNewsLayout()
        {
            //wait for stage to exist
            addEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
        }

        private function onAddedToStage(evtObj:Event):void
        {
            removeEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            // Headline text flow and flow composer
            hTextFlow = TextConverter.importToFlow(headlineMarkup,
TextConverter.TEXT_LAYOUT_FORMAT);

            // initialize the headline container and controller objects
            headContainer = new Sprite();
            headlineController = new ContainerController(headContainer);
            headlineController.verticalScrollPolicy = ScrollPolicy.OFF;
            hContainerFormat = new TextLayoutFormat();
            hContainerFormat.paddingTop = 4;
            hContainerFormat.paddingRight = 4;
```

```
            hContainerFormat.paddingBottom = 4;
            hContainerFormat.paddingLeft = 4;

            headlineController.format = hContainerFormat;
            hTextFlow.flowComposer.addController(headlineController);
            addChild(headContainer);
            stage.addEventListener(flash.events.Event.RESIZE, resizeHandler);

            // Body text TextFlow and flow composer
            bTextFlow = TextConverter.importToFlow(bodyMarkup,
TextConverter.TEXT_LAYOUT_FORMAT);

            // The body text container is below, and has three columns
            bodyTextContainer = new Sprite();
            bodyController = new ContainerController(bodyTextContainer);
            bodyTextContainerFormat = new TextLayoutFormat();
            bodyTextContainerFormat.columnCount = 3;
            bodyTextContainerFormat.columnGap = 30;

            bodyController.format = bodyTextContainerFormat;
            bTextFlow.flowComposer.addController(bodyController);
            addChild(bodyTextContainer);
            resizeHandler(null);
        }

        private function resizeHandler(event:Event):void
        {
            const verticalGap:Number = 25;
            const stagePadding:Number = 16;
            var stageWidth:Number = stage.stageWidth - stagePadding;
            var stageHeight:Number = stage.stageHeight - stagePadding;
            var headlineWidth:Number = stageWidth;
            var headlineContainerHeight:Number = stageHeight;

            // Initial compose to get height of headline after resize
            headlineController.setCompositionSize(headlineWidth,
headlineContainerHeight);
            hTextFlow.flowComposer.compose();
            var rect:Rectangle = headlineController.getContentBounds();
            headlineContainerHeight = rect.height;
```

```
            // Resize and place headline text container
            // Call setCompositionSize() again with updated headline height
            headlineController.setCompositionSize(headlineWidth, headlineContainerHeight );
            headlineController.container.x = stagePadding / 2;
            headlineController.container.y = stagePadding / 2;
            hTextFlow.flowComposer.updateAllControllers();

            // Resize and place body text container
            var bodyContainerHeight:Number = (stageHeight - verticalGap -
headlineContainerHeight);
            bodyController.format = bodyTextContainerFormat;
            bodyController.setCompositionSize(stageWidth, bodyContainerHeight );
            bodyController.container.x = (stagePadding/2);
            bodyController.container.y = (stagePadding/2) + headlineContainerHeight +
verticalGap;
            bTextFlow.flowComposer.updateAllControllers();
        }
    }
}
```

Die TLFNewsLayout-Klasse verwendet zwei Textcontainer. Ein Container zeigt die Schlagzeile und die Überschrift, der andere Container zeigt den aus drei Spalten bestehenden Text. Der Einfachheit halber ist der Text im Beispiel als TLF-Markuptext fest programmiert. Die `headlineMarkup`-Variable enthält sowohl die Schlagzeile als auch die Überschrift und die `bodyMarkup`-Variable enthält den Haupttext. Weitere Informationen zu TLF-Markup finden Sie unter „Strukturieren von Text mit dem TLF“ auf Seite 462.

Nach der Initialisierung importiert die `onAddedToStage()`-Funktion den Text der Schlagzeile in ein TextFlow-Objekt, das die Hauptdatenstruktur des TLF bildet:

```
hTextFlow = TextConverter.importToFlow(headlineMarkup, TextConverter.TEXT_LAYOUT_FORMAT);
```

Dann wird ein Sprite-Objekt für den Container erstellt und ein Controller wird erstellt und dem Container zugeordnet.

```
headContainer = new Sprite();
headlineController = new ContainerController(headContainer);
```

Der Controller wird initialisiert, um Optionen für Formatierung, Bildlauf und andere Aspekte festzulegen. Der Controller enthält Geometrie, die die Grenzen des Containers für den Textfluss definiert. Ein TextLayoutFormat-Objekt enthält die folgenden Formatierungsoptionen:

```
hContainerFormat = new TextLayoutFormat();
```

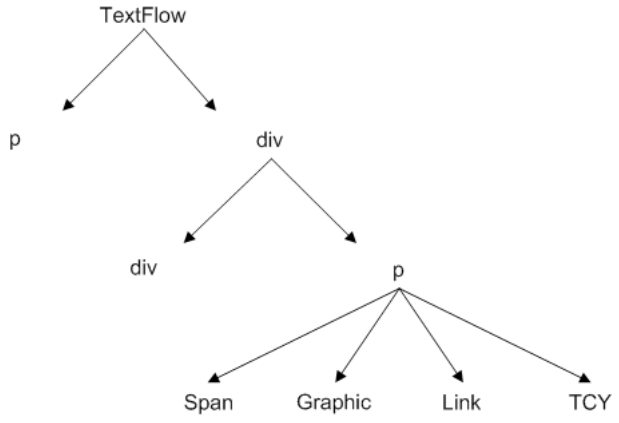
Der Controller wird dem Flow-Composer zugewiesen und die Funktion fügt den Container der Anzeigeliste hinzu. Der eigentliche Satz und die Anzeige der Container werden an die `resizeHandler()`-Methode übertragen. Dieselbe Schrittfolge wird ausgeführt, um das TextFlow-Objekt für den Haupttext zu initialisieren.

Die `resizeHandler()`-Methode misst den verfügbaren Platz für die Wiedergabe der Container und stellt die Größe der Container entsprechend ein. Ein anfänglicher Aufruf der `compose()`-Methode ermöglicht die Berechnung der richtigen Höhe für den Schlagzeilencontainer. Die `resizeHandler()`-Methode kann den Schlagzeilencontainer dann mit der `updateAllControllers()`-Methode platzieren und anzeigen. Zum Schluss verwendet die `resizeHandler()`-Methode die Größe des Schlagzeilencontainers, um die Position des Textkörpercontainers zu bestimmen.

Strukturieren von Text mit dem TLF

Das TLF verwendet einen Hierarchiebaum zur Darstellung des Textes. Jeder Knoten des Baums ist jeweils eine Instanz einer Klasse, die im Paket der Elemente definiert ist. Zum Beispiel: Der Stammknoten des Baums ist immer eine Instanz der TextFlow-Klasse. Die TextFlow-Klasse stellt einen gesamten Artikel dar. Ein Artikel ist eine Sammlung aus Text und anderen Elementen, die als eine Einheit oder ein Textfluss betrachtet wird. Für die Anzeige eines Artikels sind möglicherweise mehrere Spalten oder Textcontainer erforderlich.

Mit Ausnahme des Stammknotens basieren die restlichen Elemente frei auf XHTML-Elementen. Im folgenden Diagramm wird die Framework-Hierarchie dargestellt:



TextFlow-Hierarchie

TLF-Markup

Das Verständnis der TLF-Struktur ist auch beim Umgang mit TLF-Markup hilfreich. TLF-Markup ist eine XML-Darstellung von Text, die Teil des TLF ist. Zwar unterstützt das Framework auch andere XML-Formate, doch das TLF-Markup ist einzigartig, da es speziell auf der Struktur der TextFlow-Hierarchie basiert. Wenn Sie XML aus einem TextFlow mit diesem Markup-Format exportieren, erfolgt der XML-Export mit der intakten Hierarchie.

TLF-Markup stellt den Text in einer TextFlow-Hierarchie mit höchster Genauigkeit und Detailtreue dar. Die Markupsprache bietet Tags für alle Grundelemente der TextFlow-Hierarchie sowie Attribute für alle Formatierungseigenschaften, die in der TextLayoutFormat-Klasse zur Verfügung stehen.

In der folgenden Tabelle sind die Tags aufgeführt, die in TLF-Markup verwendet werden können.

Element	Beschreibung	Untergeordnete Elemente	Klasse
textflow	Das Markup-Stammelement.	div, p	TextFlow
div	Ein Abschnitt in einem TextFlow. Kann eine Gruppe von Absätzen enthalten.	div, list, p	DivElement
p	Ein Absatz.	a, tcy, span, img, tab, br, g	ParagraphElement
a	Ein Hyperlink.	tcy, span, img, tab, br, g	LinkElement
tcy	Eine horizontale Textfolge (wird in einem vertikalen TextFlow-Objekt verwendet).	a, span, img, tab, br, g	TCYElement
span	Eine Textfolge innerhalb eines Absatzes.		SpanElement

Element	Beschreibung	Untergeordnete Elemente	Klasse
img	Ein Bild in einem Absatz.		InlineGraphicElement
tab	Ein Tabulatorzeichen.		TabElement
br	Ein Zeilenumbruchzeichen. Wird für das Ende einer Zeile in einem Absatz verwendet; der Text wird in der nächsten Zeile fortgeführt, bleibt aber in demselben Absatz.		BreakElement
linkNormalFormat	Definiert die Formatierungsattribute für Links im normalen Zustand.	TextLayoutFormat	TextLayoutFormat
linkActiveFormat	Definiert die Formatierungsattribute, die für aktive Links verwendet werden, wenn mit der Maus auf einen Link geklickt wird.	TextLayoutFormat	TextLayoutFormat
linkHoverFormat	Definiert die Formatierungsattribute, die für Links verwendet werden, über die der Mauszeiger bewegt wird („Hover"-Zustand).	TextLayoutFormat	TextLayoutFormat
li	Ein Listeneintragselement. Muss sich in einem Listenelement befinden	div, li, list, p	ListItemElement
list	Eine Liste. Listen können verschachtelt oder nebeneinander platziert werden. Verschiedene Beschriftungen oder Nummerierungsschemas können auf die Listenelemente angewendet werden.	div, li, list, p	ListElement
g	Ein Gruppenelement. Zum Gruppieren von Elementen in einem Absatz. Hiermit können Sie Elemente unter der Absatzebene verschachteln.	a, tcy, span, img, tab, br, g	SubParagraphGroupElement

Verwandte Hilfethemen

TLF 2.0 Listen-Markup

TLF 2.0 SubParagraphGroupElements und typeName

Verwenden von Listen mit Nummern und Aufzählungszeichen

Mit den ListElement- und ListItemElement-Klassen können Sie Ihren Textsteuerelementen Listen mit Aufzählungszeichen hinzufügen. Die Listen mit Aufzählungszeichen können verschachtelt und angepasst werden. Beispielsweise können Sie verschiedene Aufzählungszeichen (oder Markierungen) und automatische Nummerierungen sowie eine Nummerierung mit Gliederung verwenden.

Zum Erstellen von Listen im Textfluss verwenden Sie das `<list>`-Tag. Dann verwenden Sie für jedes Listenelement in der Liste `<li>`-Tags innerhalb des `<list>`-Tags. Mithilfe der ListMarkerFormat-Klasse können Sie das Erscheinungsbild der Aufzählungszeichen anpassen.

Im folgenden Beispiel werden einfache Listen erstellt:

```
<flow:list paddingRight="24" paddingLeft="24">
    <flow:li>Item 1</flow:li>
    <flow:li>Item 2</flow:li>
    <flow:li>Item 3</flow:li>
</flow:list>
```

Sie können Listen in anderen Listen verschachteln, wie im folgenden Beispiel gezeigt:

```
<flow:list paddingRight="24" paddingLeft="24">
    <flow:li>Item 1</flow:li>
    <flow:list paddingRight="24" paddingLeft="24">
        <flow:li>Item 1a</flow:li>
        <flow:li>Item 1b</flow:li>
        <flow:li>Item 1c</flow:li>
    </flow:list>
    <flow:li>Item 2</flow:li>
    <flow:li>Item 3</flow:li>
</flow:list>
```

Zum Anpassen der Markierung in der Liste verwenden Sie die `listStyleType`-Eigenschaft des ListElement-Objekts. Diese Eigenschaft kann einen der Werte haben, die von der ListStyleType-Klasse definiert werden (wie `check`, `circle`, `decimal` und `box`). Das folgende Beispiel erstellt Listen mit verschiedenen Markierungstypen und einem benutzerdefinierten Wert zum Erhöhen des Zählers:

```
<flow:list paddingRight="24" paddingLeft="24" listStyleType="upperAlpha">
<flow:li>upperAlpha item</flow:li> <flow:li>another</flow:li> </flow:list> <flow:list
paddingRight="24" paddingLeft="24" listStyleType="lowerAlpha"> <flow:li>lowerAlpha
item</flow:li> <flow:li>another</flow:li> </flow:list> <flow:list paddingRight="24"
paddingLeft="24" listStyleType="upperRoman"> <flow:li>upperRoman item</flow:li>
<flow:li>another</flow:li> </flow:list> <flow:list paddingRight="24" paddingLeft="24"
listStyleType="lowerRoman"> <flow:listMarkerFormat> <!-- Increments the list by 2s rather than
1s. --> <flow:ListMarkerFormat counterIncrement="ordered 2"/> </flow:listMarkerFormat>
<flow:li>lowerRoman item</flow:li> <flow:li>another</flow:li> </flow:list>
```

Sie definieren den Zähler mithilfe der ListMarkerFormat-Klasse. Sie können nicht nur den Wert zum Erhöhen des Zählers definieren, sondern den Zähler auch anpassen, indem Sie ihn mit der `counterReset`-Eigenschaft zurücksetzen.

Das Erscheinungsbild der Markierungen in Listen kann über die `beforeContent`- und `afterContent`-Eigenschaften von ListMarkerFormat noch weiter angepasst werden. Diese Eigenschaften gelten für Inhalt, der vor und nach dem Inhalt der Markierung steht.

Im folgenden Beispiel wird der String „XX“ vor der Markierung hinzugefügt und der String „YY“ nach der Markierung:

```
<flow:list listStyleType="upperRoman" paddingLeft="36" paddingRight="24">
    <flow:listMarkerFormat>
        <flow:ListMarkerFormat fontSize="16"
            beforeContent="XX"
            afterContent="YY"
            counterIncrement="ordered -1"/>
        </flow:listMarkerFormat>
    <flow:li>Item 1</flow:li>
    <flow:li>Item 2</flow:li>
    <flow:li>Item 3</flow:li>
</flow:list>
```

Mit der `content`-Eigenschaft selbst können weitere Anpassungen des Markierungsformats vorgenommen werden. Das folgende Beispiel zeigt eine geordnete Markierung mit römischen Zahlen in Großbuchstaben.

```
<flow:list listStyleType="disc"  paddingLeft="96" paddingRight="24">
    <flow:listMarkerFormat>
        <flow:ListMarkerFormat fontSize="16"
            beforeContent="Section "
            content="counters(ordered,&quot;*&quot;,upperRoman)"
            afterContent=": "/>
    </flow:listMarkerFormat>
    <flow:li>Item 1</li>
    <flow:li>Item 2</li>
    <flow:li>Item 3</li>
</flow:list>
```

Wie das vorherige Beispiel zeigt, kann die `content`-Eigenschaft auch ein Suffix einfügen. Dies ist ein String, der nach der Markierung, aber vor `afterContent` steht. Um diesen String einzufügen, wenn XML-Inhalt für den Fluss angegeben wird, umschließen Sie den String in `&quote;`-HTML-Entitäten anstatt in Anführungszeichen (`"<String>"`).

Verwandte Hilfethemen

TLF 2.0 Listen-Markup

Auffüllen in TLF

Jedes FlowElement unterstützt padding-Eigenschaften zum Auffüllen, mit denen Sie die Position des Inhaltsbereichs der einzelnen Elemente und den Platz zwischen den Inhaltsbereichen steuern können.

Die Gesamtbreite eines Elements setzt sich zusammen aus der Breite des Inhalts sowie den `paddingLeft`- und `paddingRight`-Eigenschaften. Die Gesamthöhe des Elements besteht aus der Höhe des Inhalts sowie den `paddingTop`- und `paddingBottom`-Eigenschaften.

Die Auffüllung ist der Platz zwischen dem Rahmen und dem Inhalt. Die Eigenschaften zum Auffüllen heißen `paddingBottom`, `paddingTop`, `paddingLeft` und `paddingRight`. Die Auffüllung kann auf das TextFlow-Objekt sowie die folgenden untergeordneten Elemente angewendet werden:

- div
- img
- li
- list
- p

Eigenschaften zum Auffüllen können nicht auf span-Elemente angewendet werden.

Im folgenden Beispiel werden die Auffülleigenschaften für TextFlow festgelegt:

```
<flow:TextFlow version="2.0.0" xmlns:flow="http://ns.adobe.com/textLayout/2008" fontSize="14"
textIndent="15" paddingTop="4" paddingLeft="4" fontFamily="Times New Roman">
```

Gültige Werte für die Auffülleigenschaften sind eine Zahl (in Pixeln), „auto“ und „inherit“. Der Standardwert „auto“ wird automatisch berechnet und bei allen Elementen mit Ausnahme von ListElement auf 0 eingestellt. Bei ListElement-Objekten entspricht „auto“ 0, mit Ausnahme der Startseite der Liste, wo der Wert der `listAutoPadding`-Eigenschaft verwendet wird. Der Standardwert von `listAutoPadding` ist 40, wodurch Listen mit einem Standardeinzug versehen werden.

Standardmäßig erben die Auffülleigenschaften keine Werte. Die Werte „auto“ und „inherit“ sind Konstanten, die von der FormatValue-Klasse definiert werden.

Auffülleigenschaften können negative Werte aufweisen.

Verwandte Hilfethemen

Änderungen an der Auffüllung in TLF 2.0

Formatieren von Text mit dem TLF

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Das flashx.textLayout.formats-Paket enthält Schnittstellen und Klassen, die es Ihnen ermöglichen, jedem FlowElement in der Textflusshierarchie Formate zuzuweisen. Es gibt zwei Möglichkeiten, Formatierung anzuwenden. Sie können ein bestimmtes Format einzeln zuweisen oder mit einem speziellen Formatierungsobjekt eine Gruppe von Formaten gleichzeitig zuweisen.

Die ITextLayoutFormat-Schnittstelle enthält alle Formate, die auf ein FlowElement angewendet werden können. Manche Formate gelten für einen gesamten Container oder Textabsatz, aber nicht logischerweise für einzelne Zeichen. Zum Beispiel gelten Formate wie Ausrichtung oder Tabulatoren für ganze Absätze, können jedoch nicht auf einzelne Zeichen angewendet werden.

Zuweisen von Formaten zu einem FlowElement mit Eigenschaften

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können Formate für ein FlowElement über Eigenschaftszuweisungen festlegen. Die FlowElement-Klasse implementiert die ITextLayoutFormat-Schnittstelle; deshalb muss jede Unterklasse der FlowElement-Klasse diese Schnittstelle ebenfalls implementieren.

Der folgende Code zeigt zum Beispiel, wie einzelne Formate einer ParagraphElement-Instanz zugewiesen werden:

```
var p:ParagraphElement = new ParagraphElement();
p.fontSize = 18;
p.fontFamily = "Arial";
```

Zuweisen von Formaten zu einem FlowElement mit der TextLayoutFormat-Klasse

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie können mit der TextLayoutFormat-Klasse Formate auf ein FlowElement anwenden. Sie verwenden diese Klasse zur Erstellung eines besonderen Formatierungsobjekts, das alle gewünschten Formatierungswerte enthält. Anschließend können Sie dieses Objekt der `format`-Eigenschaft eines beliebigen FlowElement-Objekts zuweisen. Sowohl TextLayoutFormat als auch FlowElement implementieren die ITextLayoutFormat-Schnittstelle. Auf diese Weise wird sichergestellt, dass beide Klassen dieselben Formateigenschaften enthalten.

Weitere Informationen finden Sie im Abschnitt zu TextLayoutFormat im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Formatvererbung

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Formate werden über die Textflusshierarchie vererbt. Wenn Sie einer FlowElement-Instanz mit untergeordneten Elementen eine TextLayoutFormat-Instanz zuweisen, initiiert das Framework einen Prozess, der *Kaskade* genannt wird. Während einer Kaskade untersucht das Framework jeden Knoten in der Hierarchie, der Ihr FlowElement beerbt. Danach wird bestimmt, ob die vererbten Werte jeder Formatierungseigenschaft zugewiesen werden. Die folgenden Regeln werden während einer Kaskade angewandt:

1. Eigenschaftswerte können nur von direkt übergeordneten Elementen geerbt werden. Diese Elemente nennt man auch Parents.
2. Eigenschaftswerte können nur vererbt werden, wenn eine Eigenschaft noch keinen Wert besitzt (d. h. der Wert ist `undefined`).
3. Einige Attribute erben keine Werte, wenn sie undefiniert sind, es sei denn, der Attributwert ist auf „inherit“ oder auf die Konstante `flashx.textLayout.formats.FormatValue.INHERIT` gesetzt.

Wenn Sie zum Beispiel den `fontSize`-Wert auf der TextFlow-Ebene setzen, gilt dieser für alle Elemente im TextFlow. Mit anderen Worten, die Werte werden stufenweise die TextFlow-Hierarchie heruntervererbt. Sie können jedoch den Wert in einem bestimmten Element außer Kraft setzen, indem Sie dem Element direkt einen neuen Wert zuweisen. Wenn Sie dagegen den `backgroundColor`-Wert auf der TextFlow-Ebene festlegen, erben die untergeordneten Elemente von TextFlow diesen Wert nicht. Die `backgroundColor`-Eigenschaft erbt in einer Kaskade nicht von ihrem übergeordneten Element. Sie können dieses Verhalten außer Kraft setzen, indem Sie die `backgroundColor`-Eigenschaft jedes untergeordneten Elements auf `flashx.textLayout.formats.FormatValue.INHERIT` setzen.

Weitere Informationen finden Sie im Abschnitt zu TextLayoutFormat im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Importieren und Exportieren mit TLF

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Mit der TextConverter-Klasse im flashx.textLayout.conversion.*-Paket können Sie Text in das TLF importieren und aus dem TLF exportieren. Verwenden Sie diese Klasse, wenn Sie den Text zur Laufzeit laden möchten, anstatt ihn in der SWF-Datei zu kompilieren. Sie können diese Klasse auch verwenden, um Text, der in einer TextFlow-Instanz gespeichert ist, in ein String- oder XML-Objekt zu exportieren.

Sowohl Importieren als auch Exportieren sind einfache Prozesse. Rufen Sie dazu entweder die `export()`- oder `importToFlow()`-Methode auf, wobei beide Teil der TextConverter-Klasse sind. Beides sind statische Methoden, d. h. dass Sie die Methoden in der TextConverter-Klasse aufrufen und nicht in einer Instanz der TextConverter-Klasse.

Die Klassen im flashx.textLayout.conversion-Paket bieten Ihnen bei der Entscheidung, wo Sie Text speichern möchten, ein hohes Maß an Flexibilität. Wenn Sie Ihren Text beispielsweise in einer Datenbank speichern, können Sie ihn zur Anzeige in das Framework importieren. Dann können Sie die Klassen im flashx.textLayout.edit-Paket verwenden, um den Text zu ändern und den geänderten Text wieder in die Datenbank zu exportieren.

Weitere Informationen finden Sie im Abschnitt zu flashx.textLayout.conversion im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Verwalten von Textcontainern mit dem TLF

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Nachdem Text in TLF-Datenstrukturen gespeichert wurde, kann er in Flash Player angezeigt werden. Text, der in der Flusshierarchie gespeichert wurde, muss in ein Format umgewandelt werden, das Flash Player anzeigen kann. Das TLF bietet zwei Verfahren zum Erstellen von Anzeigeobjekten anhand eines Flusses. Das erste und einfachere Verfahren eignet sich für die Anzeige von statischem Text. Das zweite Verfahren ist etwas komplizierter, ermöglicht es Ihnen aber, dynamischen Text zu erstellen, der ausgewählt und bearbeitet werden kann. In beiden Fällen wird der Text letztendlich in Instanzen der TextLine-Klasse umgewandelt, die im flash.text.engine.*-Paket von Flash Player 10 enthalten ist.

Erstellen von statischem Text

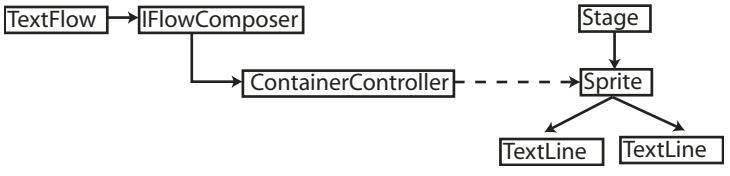
Bei diesem einfachen Ansatz wird die TextFlowTextLineFactory-Klasse verwendet, die sich im flashx.textLayout.factory-Paket befindet. Der Vorteil dieser Variante ist, dass sie nicht nur einfach ist, sondern auch weniger Direktzugriffsspeicher als der FlowComposer-Ansatz benötigt. Dieser Ansatz eignet sich für statischen Text, den der Benutzer nicht bearbeiten, auswählen oder scrollen muss.

Weitere Informationen finden Sie im Abschnitt zu TextFlowTextLineFactory im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Erstellen von dynamischem Text und Containern

Verwenden Sie einen Flow-Composer, wenn Sie die Textanzeige genauer steuern möchten, als dies mit TextFlowTextLineFactory möglich ist. Ein Flow-Composer ermöglicht es den Benutzern beispielsweise, Text auszuwählen und zu bearbeiten. Weitere Informationen finden Sie unter „Aktivieren von Textauswahl, Bearbeitung und Rückgängigmachen mit dem TLF“ auf Seite 469.

Ein Flow-Composer ist eine Instanz der StandardFlowComposer-Klasse im flashx.textLayout.compose-Paket. Ein Flow-Composer ist für die Umwandlung von TextFlow- in TextLine-Instanzen zuständig. Außerdem steuert er die Platzierung dieser TextLine-Instanzen in einem oder mehreren Containern.



Ein IFlowComposer hat keine oder mehrere ContainerController.

Jede TextFlow-Instanz hat ein entsprechendes Objekt, das die IFlowComposer-Schnittstelle implementiert. Auf dieses IFlowComposer-Objekt wird über die `TextFlow.flowComposer`-Eigenschaft zugegriffen. Sie können Methoden, die über die IFlowComposer-Schnittstelle definiert sind, über diese Eigenschaft aufrufen. Mithilfe dieser Methoden können Sie den Text einem oder mehreren Containern zuordnen und den Text für die Anzeige in einem Container vorbereiten.

Ein Container ist eine Instanz der Sprite-Klasse, die eine Unterklasse der DisplayObjectContainer-Klasse ist. Beide Klassen sind Bestandteil der Anzeigelisten-API von Flash Player. Ein Container ist eine komplexere Form des Begrenzungsrechtecks, das mit der TextLineFactory-Klasse verwendet wird. Wie das Begrenzungsrechteck definiert ein Container den Bereich, in dem TextLine-Instanzen dargestellt werden. Im Gegensatz zu einem Begrenzungsrechteck hat ein Container ein entsprechendes „Controller"-Objekt. Der Controller verwaltet Bildlauf, Satz, Verknüpfung, Formatierung und Ereignisverarbeitung für einen oder mehrere Container. Jeder Container verfügt über ein entsprechendes Controllerobjekt, das eine Instanz der ContainerController-Klasse im flashx.textLayout.container-Paket ist.

Zum Anzeigen von Text erstellen Sie ein Controllerobjekt, das den Container verwaltet, und verknüpfen Sie es mit dem Flow-Composer. Nachdem Sie den Container verknüpft haben, müssen Sie den Text zusammenstellen, damit Sie ihn anzeigen können. Dementsprechend haben Container zwei Zustände: Satz (Composition) und Anzeige. „Satz" bezeichnet den Prozess, bei dem der Text aus der Textflusshierarchie in TextLine-Instanzen konvertiert und berechnet wird, ob letztere im Container Platz haben. „Anzeige" ist der Prozess, bei dem die Anzeigeliste von Flash Player aktualisiert wird.

Weitere Informationen finden Sie unter IFlowComposer, StandardFlowComposer und ContainerController im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Aktivieren von Textauswahl, Bearbeitung und Rückgängigmachen mit dem TLF

Flash Player 9.0 und höher, Adobe AIR 1.0 und höher

Die Möglichkeit, Text auszuwählen oder zu bearbeiten, wird auf Textflussebene gesteuert. Jede Instanz der TextFlow-Klasse hat einen zugeordneten Interaktionsmanager. Sie können über die `TextFlow.interactionManager`-Eigenschaft eines TextFlow-Objekts auf dessen Interaktionsmanager zugreifen. Um die Textauswahl zu aktivieren, weisen Sie der `interactionManager`-Eigenschaft eine Instanz der SelectionManager-Klasse zu. Um sowohl die Textauswahl als auch die Bearbeitung zu aktivieren, weisen Sie eine Instanz der EditManager-Klasse anstelle einer Instanz der SelectionManager-Klasse zu. Um Rückgängig-Vorgänge zu aktivieren, erstellen Sie eine Instanz der UndoManager-Klasse und schließen Sie sie als Argument ein, wenn Sie den Konstruktor für EditManager aufrufen. Die UndoManager-Klasse speichert einen Verlauf der neuesten Bearbeitungsaktivitäten des Benutzers und stellt dem Benutzer für bestimmte Bearbeitungen die Funktionen „Rückgängig" und „Wiederholen" zur Verfügung. Diese drei Klassen sind Teil des edit-Pakets.

Weitere Informationen finden Sie unter SelectionManager, EditManager und UndoManager im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Ereignisverarbeitung mit dem TLF

Flash Player 10 und höher, Adobe AIR 1.5 und höher

TextFlow-Objekte lösen in zahlreichen Situationen Ereignisse aus, wie zum Beispiel:

- Bei einer Änderung des Textes oder des Layouts
- Vor Beginn und nach Abschluss eines Vorgangs
- Bei einer Statusänderung eines FlowElement-Objekts

- Bei Abschluss einer Zusammenstellung

Weitere Informationen finden Sie im Abschnitt zu flashx.textLayout.events im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Verwandte Hilfethemen

TLF FlowElement- und LinkElement-Ereignisse und EventMirrors

Positionieren von Bildern innerhalb von Text

Zum Positionieren von InlineGraphicElement-Objekten innerhalb des Textes verwenden Sie die folgenden Eigenschaften:

- `float`-Eigenschaft der InlineGraphicElement-Klasse
- `clearFloats`-Eigenschaft von FlowElement

Die `float`-Eigenschaft steuert die Platzierung der Grafik und des umgebenden Textes. Die `clearFloats`-Eigenschaft steuert die Platzierung der Absatzelemente relativ zu `float`.

Um die Position eines Bildes innerhalb eines Textelements zu steuern, verwenden Sie die `float`-Eigenschaft. Im folgenden Beispiel wird ein Bild einem Absatz hinzugefügt und links ausgerichtet, damit der Text rechts umgebrochen wird.

```
<flow:p paragraphSpaceAfter="15" >Images in a flow are a good thing. For example, here is a
float. It should show on the left: <flow:img float="left" height="50" width="19"
source="../assets/bulldog.jpg"></flow:img> Don't you agree? Another sentence here. Another
sentence here. Another sentence here. Another sentence here. Another sentence here. Another
sentence here. Another sentence here. Another sentence here.</flow:p>
```

Gültige Werte für die `float`-Eigenschaft sind: left, right, start, end, none. Die Float-Klasse definiert diese Konstanten. Der Standardwert lautet „none“.

Die `clearFloats`-Eigenschaft eignet sich, wenn Sie die Startposition von folgenden Absätzen anpassen möchten, die andernfalls das Bild umgeben würden. Angenommen, Ihr Bild ist größer als der erste Absatz. Um sicherzustellen, dass der zweite Absatz *nach* dem Bild beginnt, setzen Sie die `clearFloats`-Eigenschaft.

Im folgenden Beispiel ist das Bild höher als der Text im ersten Absatz. Damit der zweite Absatz nach dem Bild im Textblock beginnt, wird in diesem Beispiel die `clearFloats`-Eigenschaft für den zweiten Absatz auf „end“ gesetzt.

```
<flow:p paragraphSpaceAfter="15" >Here is another float, it should show up on the right:
<flow:img float="right" height="50" elementHeight="200" width="19"
source="../assets/bulldog.jpg"></flow:img>We'll add another paragraph that should clear past
it.</flow:p><flow:p clearFloats="end" >This should appear after the previous float on the
right.</flow:p>
```

Gültige Werte für die `clearFloats`-Eigenschaft sind: left, right, end, start, none, both. Die ClearFloats-Klasse definiert diese Konstanten. Sie können die `clearFloats`-Eigenschaft auch auf „inherit“ setzen; dies ist eine Konstante, die von der FormatValue-Klasse definiert wird. Der Standardwert lautet „none“.

Verwandte Hilfethemen

TLF-Floats

Kapitel 24: Arbeiten mit Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript wurde für den Benutzer einbeziehende, interaktive Anwendungen konzipiert – und ein häufig übersehenes Element für den Erfolg solcher Anwendungen ist der Sound. Mit Sound als Schwerpunkt einer Anwendung können Sie einem Videospiel Soundeffekte hinzufügen, der Benutzeroberfläche einer Anwendung ein Audio-Feedback hinzufügen oder sogar ein Programm erstellen, das über das Internet geladene MP3-Dateien analysiert.

Sie können externe Audiodateien laden und mit Audio arbeiten, das in eine SWF-Datei eingebettet ist. Sie können die Audiowiedergabe steuern, visuelle Darstellungen der Sounddaten erstellen und Sound über ein Benutzermikrofon erfassen.

Verwandte Hilfethemen

flash.media-Paket

flash.events.SampleDataEvent

Grundlagen zum Arbeiten mit Sound

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Computer können digitalen Sound - eine Computerdarstellung der Sounddaten - erfassen, umwandeln, speichern und zur Wiedergabe über die an den Computer angeschlossenen Lautsprecher abrufen. Sound können Sie mithilfe von Adobe® Flash® Player oder Adobe® AIR™ und ActionScript wiedergeben.

Sounddaten, die in ein digitales Format umgewandelt wurden, weisen verschiedene Eigenschaften auf, z. B. die Lautstärke des Sounds und ob der Sound in Mono oder Stereo vorliegt. Wenn Sie einen Sound in ActionScript wiedergeben, können Sie diese Eigenschaften ebenfalls einstellen – z. B. können Sie den Sound lauter stellen oder den Eindruck erwecken, als käme er aus einer bestimmten Richtung.

Bevor Sie einen Sound mit ActionScript steuern können, müssen Sie die Soundinformationen in Flash Player oder AIR laden. Es gibt fünf Möglichkeiten, wie Sie Audiodaten in Flash Player oder AIR laden können, sodass Sie diese Daten mit ActionScript bearbeiten können.

- Laden einer externen Sounddatei in die SWF, wie eine MP3-Datei.
- Einbetten der Sounddaten in die SWF-Datei direkt bei der Erstellung.
- Aufnehmen von Audio über ein Mikrofon, das an den Computer des Benutzers angeschlossen ist.
- Streamen von Audio von einem Server.
- Dynamisches Erstellen und Abspielen von Audio.

Wenn Sie die Sounddaten von einer externen Sounddatei laden, können Sie die Wiedergabe der Sounddatei starten, noch bevor die Daten vollständig geladen sind.

Obwohl es verschiedene Dateiformate gibt, die zur Kodierung von Audiodaten verwendet werden, unterstützen ActionScript 3.0, Flash Player und AIR nur Sounddateien, die im MP3-Format gespeichert sind. Sie können keine Sounddateien in anderen Formaten wie WAV oder AIFF laden oder wiedergeben.

Beim Verwenden von Sound in ActionScript werden Sie wahrscheinlich verschiedene Klassen aus dem flash.media-Paket verwenden. Mithilfe der Sound-Klasse erhalten Sie Zugriff auf Audioinformationen, indem eine Sounddatei geladen oder eine Funktion einem Ereignis zugewiesen wird, das Sounddaten sampelt und dann die Wiedergabe startet. Nachdem Sie die Wiedergabe eines Sounds gestartet haben, gewähren Ihnen Flash Player und AIR den Zugriff auf ein SoundChannel-Objekt. Da eine von Ihnen geladene Audiodatei nur einer von vielen Sounds sein kann, die Sie auf einem Benutzercomputer wiedergeben, verwendet jeder einzelne Sound sein eigenes SoundChannel-Objekt. Das, was tatsächlich über die Lautsprecher des Computers wiedergegeben wird, ist die kombinierte und gemischte Ausgabe aller SoundChannel-Objekte. Mit dieser SoundChannel-Instanz können Sie die Eigenschaften des Sounds steuern und dessen Wiedergabe anhalten. Wenn Sie die kombinierten Audiodaten steuern möchten, erhalten Sie über die SoundMixer-Klasse die Kontrolle über die gemischte Ausgabe.

Außerdem können Sie beim Verwenden von Sound in ActionScript auf einige weitere Klassen zugreifen, um spezielle Aufgaben durchzuführen. Weitere Informationen über alle soundbezogenen Klassen finden Sie unter „Soundarchitektur“ auf Seite 472.

Wichtige Konzepte und Begriffe

Im Folgenden sind wichtige Begriffe aufgeführt, die in diesem Zusammenhang verwendet werden:

Amplitude Der Abstand eines Punkts auf der Sound-Wellenform von der Null- oder Äquilibriumlinie.

Bitrate Die Datenmenge, die in einer Sekunde einer Sounddatei kodiert oder gestreamt wird. Bei MP3-Dateien wird die Bitrate in der Regel in Kilobit pro Sekunde (KBit/s) angegeben. Eine höhere Bitrate führt im Allgemeinen zu einer qualitativ höherwertigen Soundwelle.

Zwischenspeicherung Das Empfangen und Speichern der Sounddaten, bevor sie wiedergegeben werden.

mp3 MPEG-1 Audio Layer 3, oder kurz MP3, ist ein populäres Sound-Komprimierungsformat.

Panning Die Positionierung eines Audiosignals zwischen dem linken und dem rechten Kanal in einem Stereo-Soundbild.

Spitze Der höchste Punkt einer Soundwelle.

Sampling-Rate Definiert die Anzahl an Werten, die pro Sekunde aus einem analogen Audiosignal erfasst werden, um ein digitales Signal zu erzeugen. Die Sampling-Rate von standardmäßigem Compact Disc-Audio beträgt 44,1 kHz oder 44.100 Abtastungen pro Sekunde.

Streaming Die Wiedergabe der Anfangsteile einer Sound- oder Videodatei, während die Datei noch vom Server geladen wird.

Lautstärke Die Lautstärke eines Sounds.

Wellenform Die Form eines Graphen mit sich ändernden Amplituden eines Soundsignals im Zeitverlauf.

Soundarchitektur

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ihre Anwendungen können Sounddaten aus fünf Hauptquellen laden:

- Externe Sounddateien, die zur Laufzeit geladen werden
- Soundressourcen, die in die SWF-Datei der Anwendung eingebettet sind
- Sounddaten von einem Mikrofon, das an das Benutzersystem angeschlossen ist

- Sounddaten, die von einem remoten Medienserver, z. B. einem Flash Media Server, gestreamt werden
- Sounddaten, die durch die Verwendung der Ereignisprozedur `sampleData` dynamisch generiert werden

Sounddaten können vollständig geladen sein, bevor die Wiedergabe beginnt, oder sie können gestreamt werden. Streamen bedeutet, dass die Wiedergabe bereits gestartet wird, obwohl die Datei noch geladen wird.

Die Soundklassen von ActionScript 3.0 unterstützen Sounddateien, die im MP3-Format gespeichert sind. Sie können keine Sounddateien in anderen Formaten wie WAV oder AIFF direkt laden oder wiedergeben. Ab Flash Player 9.0.115.0 können jedoch mithilfe der NetStream-Klasse AAC-Audiodaten geladen und abgespielt werden. Dabei handelt es sich um die gleiche Technik, die zum Laden und Wiedergeben von Videoinhalt verwendet wird. Weitere Informationen zu dieser Technik finden Sie unter „Verwenden von Videos“ auf Seite 505.

Mit Adobe Flash Professional lassen sich Sounddateien im WAV- oder AIFF-Format importieren und dann im MP3-Format in die SWF-Dateien einer Anwendung einbetten. Mit dem Flash-Authoring-Tool können Sie auch eingebettete Sounddateien komprimieren, um die Dateigröße zu reduzieren. Dies geht jedoch auf Kosten der Soundqualität. Weitere Informationen finden Sie unter „Sounds importieren“ im Handbuch *Verwenden von Flash*.

Die ActionScript 3.0-Soundarchitektur nutzt die folgenden Klassen aus dem flash.media-Paket.

Klasse	**Beschreibung**
flash.media.Sound	Die Sound-Klasse ist für das Laden des Sounds, das Verwalten der allgemeinen Soundeigenschaften und das Starten der Soundwiedergabe verantwortlich.
flash.media.SoundChannel	Wenn eine Anwendung ein Sound-Objekt wiedergibt, wird ein neues SoundChannel-Objekte erstellt, mit dem die Wiedergabe gesteuert wird. Das SoundChannel-Objekt steuert die Lautstärke des linken und rechten Wiedergabekanals des Sounds. Jeder wiedergegebene Sound verfügt über sein eigenes SoundChannel-Objekt.
flash.media.SoundLoaderContext	Die SoundLoaderContext-Klasse gibt an, wie viele Sekunden beim Laden eines Sounds gepuffert werden, und ob Flash Player bzw. AIR beim Laden einer Datei nach einer Richtliniendatei auf dem Server sucht. Ein SoundLoaderContext-Objekt wird als Parameter für die `Sound.load()`-Methode verwendet.
flash.media.SoundMixer	Die SoundMixer-Klasse steuert die Wiedergabe- und Sicherheitseigenschaften, die für alle Sounds in einer Anwendung gelten. Tatsächlich werden mehrere Soundkanäle über ein gemeinsames SoundMixer-Objekt gemischt, daher wirken sich die Eigenschaftswerte des SoundMixer-Objekts auf alle SoundChannel-Objekte aus, die gerade wiedergegeben werden.
flash.media.SoundTransform	Die SoundTransform-Klasse enthält Werte, mit denen die Lautstärke und die Richtungseinstellung gesteuert werden. Ein SoundTransform-Objekt kann unter anderem auf ein einzelnes SoundChannel-Objekt, ein globales SoundMixer-Objekt oder ein Microphone-Objekt angewendet werden.
flash.media.ID3Info	Ein ID3Info-Objekt enthält Eigenschaften, die die häufig in MP3-Sounddateien gespeicherten ID3-Metadaten darstellen.
flash.media.Microphone	Die Microphone-Klasse stellt ein Mikrofon oder ein anderes Sound-Eingabegerät dar, das an den Benutzercomputer angeschlossen ist. Eine Audioeingabe von einem Mikrofon kann an die lokalen Lautsprecher geleitet oder an einen Remote-Server gesendet werden. Das Microphone-Objekt steuert die Signalstärke, Sampling-Rate und andere Eigenschaften des eigenen Soundstreams.
flash.media.AudioPlaybackMode	Die AudioPlaybackMode-Klasse definiert Konstanten für die `audioPlaybackMode`-Eigenschaft der SoundMixer-Klasse.

Jeder geladene und wiedergegebene Sound benötigt eine eigene Instanz der Sound- und der SoundChannel-Klasse. Die Ausgabe mehrerer SoundChannel-Instanzen wird dann während der Wiedergabe von der globalen SoundMixer-Klasse zusammengemischt.

Die Sound-, SoundChannel- und SoundMixer-Klassen werden nicht für Sounddaten verwendet, die von einem Mikrofon oder einem Streaming Media Server wie dem Flash Media Server stammen.

Laden von externen Sounddateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jede Instanz der Sound-Klasse dient dazu, eine bestimmte Soundressource zu laden und deren Wiedergabe auszulösen. Ein Sound-Objekt kann nicht von einer Anwendung wiederverwendet werden, um mehrere Sounds zu laden. Wenn die Anwendung eine neue Soundressource laden möchte, muss sie ein neues Sound-Objekt erstellen.

Wenn Sie eine kleine Sounddatei laden (beispielsweise einen Klick-Sound, der an eine Schaltfläche angehängt werden soll), kann Ihre Anwendung ein neues Sound-Objekt erstellen und dieses die Sounddatei automatisch laden lassen. Dies wird im folgenden Code gezeigt:

```
var req:URLRequest = new URLRequest("click.mp3");
var s:Sound = new Sound(req);
```

Der `Sound()`-Konstruktor benötigt als ersten Parameter ein URLRequest-Objekt. Wenn ein Wert für den URLRequest-Parameter angegeben wird, lädt das neue Sound-Objekt automatisch die angegebene Soundressource.

In allen außer in den einfachsten Fällen sollte Ihre Anwendung den Ladevorgang des Sounds überwachen und auf Fehler während des Ladens achten. Angenommen, der Klick-Sound ist relativ groß, so ist er eventuell noch nicht vollständig geladen, wenn der Benutzer auf die Schaltfläche klickt, die den Sound auslöst. Der Versuch, einen nicht geladenen Sound wiederzugeben, führt zu einem Laufzeitfehler. Bevor also Benutzer eine Aktion ausführen, die eine Soundwiedergabe auslöst, sollte gewartet werden, bis der Sound vollständig geladen ist.

Ein Sound-Objekt löst während des Ladens eines Sounds verschiedener Ereignisse aus. Ihre Anwendung kann auf diese Ereignisse überwachen, um den Ladevorgang zu verfolgen und so sicherzustellen, dass der Sound vollständig geladen ist, bevor die Wiedergabe gestartet wird. In der folgenden Tabelle sind die Ereignisse aufgeführt, die von einem Sound-Objekt ausgelöst werden können:

Ereignis	Beschreibung
open (`Event.OPEN`)	Wird direkt vor dem Beginn des Sound-Ladevorgangs ausgelöst.
progress (`ProgressEvent.PROGRESS`)	Wird regelmäßig während des Sound-Ladevorgangs ausgelöst, wenn die Daten aus einer Datei oder einem Stream empfangen werden.
id3 (`Event.ID3`)	Wird ausgelöst, wenn für einen MP3-Sound ID3-Daten zur Verfügung stehen.
complete (`Event.COMPLETE`)	Wird ausgelöst, wenn alle Daten der Soundressource geladen wurden.
ioError (`IOErrorEvent.IO_ERROR`)	Wird ausgelöst, wenn eine Sounddatei nicht gefunden werden kann oder der Ladeprozess unterbrochen wird, bevor alle Sounddaten empfangen wurden.

Im folgenden Code wird gezeigt, wie ein Sound wiedergegeben wird, nachdem er vollständig geladen wurde:

```
import flash.events.Event;
import flash.media.Sound;
import flash.net.URLRequest;

var s:Sound = new Sound();
s.addEventListener(Event.COMPLETE, onSoundLoaded);
var req:URLRequest = new URLRequest("bigSound.mp3");
s.load(req);

function onSoundLoaded(event:Event):void
{
    var localSound:Sound = event.target as Sound;
    localSound.play();
}
```

Zunächst wird in diesem Codebeispiel ein neues Sound-Objekt erstellt, ohne dass ein Anfangswert für den URLRequest-Parameter übergeben wird. Dann überwacht es auf das `Event.COMPLETE`-Ereignis vom Sound-Objekt, das die `onSoundLoaded()`-Methode ausführt, nachdem alle Sounddaten geladen wurden. Als Nächstes ruft es die `Sound.load()`-Methode mit einem neuen URLRequest-Wert für die Sounddatei auf.

Die `onSoundLoaded()`-Methode wird ausgeführt, nachdem der Sound vollständig geladen wurde. Die `target`-Eigenschaft des Event-Objekts ist ein Verweis auf das Sound-Objekt. Das Aufrufen der `play()`-Methode des Sound-Objekts startet dann die Soundwiedergabe.

Überwachen des Ladevorgangs eines Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sounddateien können sehr groß sein und benötigen dann viel Zeit, bis sie vollständig geladen sind. Da Flash Player und AIR die Möglichkeit bieten, dass eine Anwendung Sounds wiedergibt, noch bevor diese vollständig geladen sind, möchten Sie den Benutzer eventuell informieren, wie viele der Sounddaten geladen wurden und wie viel des Sounds bereits wiedergegeben wurde.

Die Sound-Klasse löst zwei Ereignisse aus, mit denen es relativ einfach ist, den Ladefortschritt eines Sounds anzuzeigen: `ProgressEvent.PROGRESS` und `Event.COMPLETE`. Im folgenden Beispiel wird gezeigt, wie Sie diese Ereignisse zum Anzeigen des Ladefortschritts eines Sounds verwenden können:

```
import flash.events.Event;
import flash.events.ProgressEvent;
import flash.media.Sound;
import flash.net.URLRequest;

var s:Sound = new Sound();
s.addEventListener(ProgressEvent.PROGRESS, onLoadProgress);
s.addEventListener(Event.COMPLETE, onLoadComplete);
s.addEventListener(IOErrorEvent.IO_ERROR, onIOError);

var req:URLRequest = new URLRequest("bigSound.mp3");
s.load(req);

function onLoadProgress(event:ProgressEvent):void
{
    var loadedPct:uint = Math.round(100 * (event.bytesLoaded / event.bytesTotal));
    trace("The sound is " + loadedPct + "% loaded.");
}

function onLoadComplete(event:Event):void
{
    var localSound:Sound = event.target as Sound;
    localSound.play();
}
function onIOError(event:IOErrorEvent)
{
    trace("The sound could not be loaded: " + event.text);
}
```

Dieser Code erstellt zunächst ein Sound-Objekt und fügt diesem dann Listener für die Ereignisse `ProgressEvent.PROGRESS` und `Event.COMPLETE` hinzu. Nachdem die `Sound.load()`-Methode aufgerufen wurde und die ersten Daten der Sounddatei empfangen wurden, tritt ein `ProgressEvent.PROGRESS`-Ereignis ein und die `onSoundLoadProgress()`-Methode wird ausgelöst.

Der Prozentsatz an geladenen Sounddaten entspricht dem Wert der `bytesLoaded`-Eigenschaft des ProgressEvent-Objekts geteilt durch den Wert der `bytesTotal`-Eigenschaft. Die gleichen `bytesLoaded`- und `bytesTotal`-Eigenschaften stehen auch für das Sound-Objekt zur Verfügung. Im vorangegangenen Beispiel werden lediglich Meldungen über den Ladefortschritt der Sounddatei angezeigt. Sie können jedoch die Werte `bytesLoaded` und `bytesTotal` auf einfache Weise integrieren, um Komponenten der Fortschrittsleiste (beispielsweise aus dem Adobe Flex-Framework oder dem Adobe Flash-Authoring-Tool) zu aktualisieren.

Dieses Beispiel zeigt auch, wie eine Anwendung einen Fehler beim Laden von Sounddateien erkennen und darauf reagieren kann. Angenommen, eine Sounddatei mit einem bestimmten Namen kann nicht geladen werden. In diesem Fall wird vom Sound-Objekt ein `Event.IO_ERROR`-Ereignis ausgelöst. Im vorangegangenen Code wird die `onIOError()`-Methode ausgeführt, die eine kurze Fehlermeldung anzeigt, wenn ein Fehler auftritt.

Verwenden eingebetteter Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eingebettete Sounds als Alternative zum Laden eines Sounds aus einer externen Datei eignen sich insbesondere für kleine Sounddateien, die als Indikatoren innerhalb der Benutzeroberfläche einer Anwendung verwendet werden, z. B. als Sounds, die beim Klicken auf Schaltflächen wiedergegeben werden.

Wenn Sie eine Sounddatei in Ihre Anwendung einbetten, wird die resultierende SWF-Datei um die Größe der Sounddatei erweitert. Anders ausgedrückt, durch das Einbetten von großen Sounddateien in Ihre Anwendung könnte Ihre SWF-Datei in nicht wünschenswerter Weise anwachsen.

Die genaue Methode des Einbettens einer Sounddatei in die SWF-Datei Ihrer Anwendung hängt von Ihrer Entwicklungsumgebung ab.

Verwenden eingebetteter Sounddateien in Flash

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem Flash-Authoring-Tool können Sie in einer Reihe von Soundformaten vorliegende Sounds importieren und als Symbole in der Bibliothek speichern. Sie können diese dann Bildern in der Zeitleiste oder Bildern eines Schaltflächenstatus zuordnen, sie zusammen mit Verhalten einsetzen oder direkt in ActionScript-Code verwenden. In diesem Abschnitt wird beschrieben, wie Sie eingebettete Sounds in ActionScript-Code mit dem Flash-Authoring-Tool verwenden können. Informationen über weitere Möglichkeiten zum Verwenden eingebetteter Sounds in Flash finden Sie unter „Sounds importieren“ im Handbuch *Verwenden von Flash*.

So betten Sie eine Sounddatei mit dem Flash-Authoring-Tool ein:

1. Klicken Sie auf „Datei“ > „Importieren“ > „In Bibliothek importieren“, und wählen Sie eine Sounddatei aus, um diese zu importieren.
2. Klicken Sie im Bedienfeld „Bibliothek“ mit der rechten Maustaste auf den Namen der importierten Datei und wählen Sie „Eigenschaften“ aus. Klicken Sie auf das Kontrollkästchen „Export für ActionScript“.
3. Geben Sie im Feld „Klasse“ einen Namen ein, der in ActionScript für Verweise auf diesen eingebetteten Sound verwendet werden soll. Standardmäßig wird in diesem Feld der Name der Sounddatei eingetragen. Wenn der Dateiname einen Punkt enthält (wie beispielsweise im Namen „DrumSound.mp3“), müssen Sie ihn entsprechend ändern (z. B. in „DrumSound“). In ActionScript sind Punkte in Klassennamen nicht zulässig. Im Feld „Basisklasse“ sollte weiterhin „flash.media.Sound“ angezeigt werden.
4. Klicken Sie auf „OK“. Möglicherweise wird ein Dialogfeld mit der Meldung angezeigt, dass im Klassenpfad keine Definition für diese Klasse gefunden wurde. Klicken Sie auf „OK“ und setzen Sie den Vorgang fort. Wenn Sie einen Klassennamen eingegeben haben, der mit keinem der Klassennamen im Klassenpfad der Anwendung übereinstimmt, wird automatisch eine neue Klasse generiert, die von der flash.media.Sound-Klasse erbt.
5. Um den eingebetteten Sound zu verwenden, nehmen Sie in ActionScript über den Klassennamen des Sounds darauf Bezug. Beispielsweise wird am Anfang des folgenden Codes eine neue Instanz der automatisch generierten DrumSound-Klasse erstellt:

   ```
   var drum:DrumSound = new DrumSound();
   var channel:SoundChannel = drum.play();
   ```

   Die DrumSound-Klasse ist eine Unterklasse der flash.media.Sound-Klasse und erbt deshalb die Methoden und Eigenschaften der Sound-Klasse, einschließlich der `play()`-Methode, wie oben dargestellt.

Verwenden eingebetteter Sounddateien in Flex

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Soundbestände können auf vielfältige Weise in eine Flex-Anwendung eingebettet werden, beispielsweise durch:

- Verwenden des Metadaten-Tags `[Embed]` in einem Skript

- Verwenden der `@Embed`-Direktive in MXML, um einen eingebetteten Bestand einer Komponente wie einem Button oder SoundEffect als Eigenschaft zuzuweisen
- Verwenden der `@Embed`-Direktive in einer CSS-Datei

In diesem Abschnitt wird das erste Verfahren beschrieben: das Einbetten von Sound in ActionScript-Code in einer Flex-Anwendung mittels des Metadaten-Tags `[Embed]`.

Verwenden Sie zum Einbetten eines Bestands in ActionScript-Code das Metadaten-Tag `[Embed]`.

Platzieren Sie die Sounddatei im Hauptquellordner oder einem anderen Ordner, der sich im Erstellungspfad Ihres Projekts befindet. Wenn der Compiler auf ein Embed-Metadaten-Tag trifft, erstellt er die eingebettete Bestandsklasse. Sie können auf die Klasse über eine Variable des Class-Datentyps zugreifen, die Sie unmittelbar nach dem `[Embed]`-Metadaten-Tag deklarieren.

Der folgende Beispielcode bettet einen Sound namens „smallSound.mp3“ ein und verwendet eine Variable namens `soundClass` zum Speichern eines Verweises auf die eingebettete Bestandsklasse, die diesem Sound zugewiesen ist. Anschließend erstellt der Code eine Instanz der eingebetteten Bestandsklasse, wandelt sie in eine Instanz der Sound-Klasse um und ruft die `play()`-Methode für diese Instanz auf:

```
package
{
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.media.SoundChannel;

    public class EmbeddedSoundExample extends Sprite
    {
        [Embed(source="smallSound.mp3")]
        public var soundClass:Class;

        public function EmbeddedSoundExample()
        {
            var smallSound:Sound = new soundClass() as Sound;
            smallSound.play();
        }
    }
}
```

Wenn Sie mithilfe des eingebetteten Sounds eine Eigenschaft einer Flex-Komponente festlegen möchten, sollten Sie den Sound in eine Instanz der mx.core.SoundAsset-Klasse anstatt in eine Instanz der Sound-Klasse umwandeln. Einen ähnlichen Beispielcode, der die SoundAsset-Klasse verwendet, finden Sie unter „Eingebettete Bestandsklassen“ im „ActionScript 3.0 – Arbeitshandbuch“.

Verwenden von Streaming-Sounddateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn eine Sound- oder Videodatei wiedergegeben wird, während die Daten noch geladen werden, wird dies als *Streaming* bezeichnet. Von einem Remote-Server geladene externe Sounddateien werden häufig gestreamt, sodass der Benutzer nicht warten muss, bis alle Sounddaten geladen sind, bevor ein Sound wiedergegeben werden kann.

Die `SoundMixer.bufferTime`-Eigenschaft steht für die Anzahl an Millisekunden der Sounddaten, die Flash Player oder AIR abrufen muss, bevor der Sound wiedergegeben wird. Anders ausgedrückt, wenn die `bufferTime`-Eigenschaft auf 5000 eingestellt ist, lädt Flash Player bzw. AIR mindestens 5000 Millisekunden lang Daten aus der Sounddatei, bevor die Wiedergabe des Sounds beginnt. Der Standardwert für die `SoundMixer.bufferTime`-Eigenschaft beträgt 1000.

Ihre Anwendung kann den globalen `SoundMixer.bufferTime`-Wert für einen bestimmten Sound außer Kraft setzen, indem Sie beim Laden des Sounds explizit einen neuen `bufferTime`-Wert angeben. Um die standardmäßige Pufferzeit außer Kraft zu setzen, erstellen Sie zunächst eine neue Instanz der SoundLoaderContext-Klasse, stellen ihre `bufferTime`-Eigenschaft ein und übergeben sie dann als einen Parameter an die `Sound.load()`-Methode. Dies wird im folgenden Code gezeigt:

```
import flash.media.Sound;
import flash.media.SoundLoaderContext;
import flash.net.URLRequest;

var s:Sound = new Sound();
var req:URLRequest = new URLRequest("bigSound.mp3");
var context:SoundLoaderContext = new SoundLoaderContext(8000, true);
s.load(req, context);
s.play();
```

Während der Sound wiedergegeben wird, versuchen Flash Player und AIR den Soundpuffer gleich groß oder größer zu halten. Wenn die Sounddaten schneller geladen als wiedergegeben werden, erfolgt die Wiedergabe unterbrechungsfrei. Wenn jedoch die Daten aufgrund von Netzwerkeinschränkungen langsamer geladen werden, könnte der Abspielkopf das Ende des Soundpuffers erreichen. In diesem Fall wird die Wiedergabe unterbrochen und automatisch fortgesetzt, sobald mehr Sounddaten geladen wurden.

Um herauszufinden, ob die Wiedergabe unterbrochen wurde, da Flash Player bzw. AIR auf Daten wartet, verwenden Sie die `Sound.isBuffering`-Eigenschaft.

Verwenden von dynamisch generierten Audiodaten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Hinweis: *Die Möglichkeit, Audio dynamisch zu generieren, ist ab Flash Player 10 und Adobe AIR 1.5 verfügbar.*

Anstatt einen vorhandenen Sound zu laden oder zu streamen, können Sie Audiodaten auch dynamisch generieren. Sie können Audiodaten erzeugen, wenn Sie einen Ereignis-Listener für das Ereignis `sampleData` eines Sound-Objekts zuweisen. (Das Ereignis `sampleData` wird in der SampleDataEvent-Klasse im Paket „flash.events“ definiert.) In dieser Umgebung lädt das Sound-Objekt keine Sounddaten aus einer Datei. Es fungiert stattdessen als Socket für Sounddaten, die zum Objekt gestreamt werden. Das Streamen erfolgt mittels einer Funktion, die Sie diesem Ereignis zuweisen.

Wenn Sie einen `sampleData`-Ereignis-Listener zu einem Sound-Objekt hinzufügen, fordert das Objekt in bestimmten Abständen Daten zum Hinzufügen zum Soundpuffer an. Dieser Puffer enthält die Daten, die das Sound-Objekt abspielt. Nachdem Sie die `play()`-Methode des Sound-Objekts aufgerufen haben, löst es beim Anfordern neuer Sounddaten das `sampleData`-Ereignis aus. (Dies gilt nur, wenn vom Sound-Objekt keine MP3-Daten aus einer Datei geladen wurden.)

Das SampleDataEvent-Objekt enthält eine `data`-Eigenschaft. In Ihrem Ereignis-Listener schreiben Sie ByteArray-Objekte in dieses `data`-Objekt. Die Byte-Arrays, die Sie in dieses Objekt schreiben, fügen zu dem gepufferten Sound weitere Daten hinzu, die vom Sound-Objekt abgespielt werden. Das Byte-Array im Puffer ist ein Stream von Gleitkommawerten von -1 bis 1. Jeder Gleitkommawert stellt eine Amplitude eines Kanals (rechts oder links) eines Soundsamples dar. Die Sounddaten werden mit einer Rate von 44.100 Samples pro Sekunde abgetastet. Jedes Sample enthält einen linken und einen rechten Kanal, die als Gleitkommadaten in einem Byte-Array mit Interleave arrangiert werden.

In Ihrer Prozedurfunktion verwenden Sie die Methode `ByteArray.writeFloat()`, um in die Eigenschaft `data` des Ereignisses `sampleData` zu schreiben. Der folgende Code erzeugt zum Beispiel eine Sinus-Kurve.

```
var mySound:Sound = new Sound();
mySound.addEventListener(SampleDataEvent.SAMPLE_DATA, sineWaveGenerator);
mySound.play();
function sineWaveGenerator(event:SampleDataEvent):void
{
    for (var i:int = 0; i < 8192; i++)
    {
        var n:Number = Math.sin((i + event.position) / Math.PI / 4);
        event.data.writeFloat(n);
        event.data.writeFloat(n);
    }
}
```

Wenn Sie `Sound.play()` aufrufen, beginnt die Anwendung, die Ereignisprozedur aufzurufen und fordert Soundsample-Daten an. Die Anwendung fährt damit fort, Ereignisse zu senden, während der Sound solange abgespielt wird, bis Sie keine Daten mehr zur Verfügung stellen oder Sie `SoundChannel.stop()` aufrufen.

Die Wartezeit des Ereignisses unterscheidet sich von Plattform zu Plattform und kann sich in zukünftigen Versionen von Flash Player und AIR ändern. Machen Sie sich nicht von einer bestimmten Wartezeit abhängig; berechnen Sie sie stattdessen selbst. Berechnen Sie die Wartezeit anhand der folgenden Formel:

```
(SampleDataEvent.position / 44.1) - SoundChannelObject.position
```

Stellen Sie zwischen 2048 und 8192 Sample für die Eigenschaft `data` des SampleDataEvent-Objekts zur Verfügung (für jeden Aufruf an den Ereignis-Listener). Um die beste Leistung zu erzielen, stellen Sie so viele Sample wie möglich (bis zu 8192) zur Verfügung. Je weniger Sample Sie zur Verfügung stellen, desto wahrscheinlicher ist es, dass während des Abspielens Klick- und Popupfenster eingeblendet werden. Dies kann von Plattform zu Plattform unterschiedlich sein und in verschiedenen Situationen auftreten, zum Beispiel wenn die Größe des Browsers verändert wird. Code, der auf einer Plattform problemlos läuft, wenn nur 2048 Samples zur Verfügung gestellt werden, tut dies auf einer anderen Plattform möglicherweise nicht. Wenn Sie die geringst mögliche Wartezeit benötigen, überlegen Sie sich, ob Sie die Menge der Daten vom Benutzer auswählen lassen.

Stellen Sie weniger als 2048 Samples (pro Aufruf an den `sampleData`-Ereignis-Listener) zur Verfügung, stoppt die Anwendung nach dem Abspielen der verbleibenden Samples. Das SoundChannel-Objekt löst dann ein SoundComplete-Ereignis aus.

Modifizieren von Sound von mp3-Daten

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Sie verwenden die Methode `Sound.extract()`, um Daten aus einem Sound-Objekt zu extrahieren. Sie können diese Daten so verwenden (und modifizieren), dass für die Wiedergabe in den dynamischen Stream eines anderen Sound-Objekts geschrieben wird. Der folgende Code verwendet zum Beispiel die Bytes einer geladenen MP3-Datei und übergibt sie über eine Filterfunktion, `upOctave()`:

```
var mySound:Sound = new Sound();
var sourceSnd:Sound = new Sound();
var urlReq:URLRequest = new URLRequest("test.mp3");
sourceSnd.load(urlReq);
sourceSnd.addEventListener(Event.COMPLETE, loaded);
function loaded(event:Event):void
{
    mySound.addEventListener(SampleDataEvent.SAMPLE_DATA, processSound);
    mySound.play();
}
function processSound(event:SampleDataEvent):void
{
        var bytes:ByteArray = new ByteArray();
        sourceSnd.extract(bytes, 8192);
        event.data.writeBytes(upOctave(bytes));
}
function upOctave(bytes:ByteArray):ByteArray
{
    var returnBytes:ByteArray = new ByteArray();
    bytes.position = 0;
    while(bytes.bytesAvailable > 0)
    {
        returnBytes.writeFloat(bytes.readFloat());
        returnBytes.writeFloat(bytes.readFloat());
        if (bytes.bytesAvailable > 0)
        {
            bytes.position += 8;
        }
    }
    return returnBytes;
}
```

Einschränkungen bei der Erzeugung von Sound

Flash Player 10 und höher, Adobe AIR 1.5 und höher

WEnn Sie einen `sampleData`-Ereignis-Listener mit einem Sound-Objekt verwenden, sind als weitere Sound-Methoden nur noch `Sound.extract()` und `Sound.play()` aktiviert. Das Aufrufen anderer Methoden oder Eigenschaften verursacht eine Ausnahme. Alle Methoden und Eigenschaften des SoundChannel-Objekts sind nach wie vor aktiviert.

Wiedergeben von Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Wiedergabe eines geladenen Sounds kann so einfach wie das Aufrufen der `Sound.play()`-Methode eines Sound-Objekts sein. Dies wird im folgenden Code gezeigt:

```
var snd:Sound = new Sound(new URLRequest("smallSound.mp3"));
snd.play();
```

Bei der Wiedergabe von Sounds mit ActionScript 3.0 können Sie die folgenden Aktionen ausführen:

- Wiedergeben eines Sounds ab einer bestimmten Startposition

- Unterbrechen eines Sounds und Fortsetzen der Wiedergabe ab der gleichen Position zu einem späteren Zeitpunkt
- Exakt wissen, wann die Wiedergabe eines Sounds beendet ist
- Den Wiedergabefortschritt eines Sounds verfolgen
- Die Lautstärke oder Richtungseinstellung ändern, während ein Sound wiedergegeben wird

Um diese Aktionen während der Wiedergabe auszuführen, sind die Klassen SoundChannel, SoundMixer und SoundTransform erforderlich.

Die SoundChannel-Klasse steuert die Wiedergabe eines bestimmten Sounds. Die `SoundChannel.position`-Eigenschaft können Sie sich als Abspielkopf vorstellen, der die aktuelle Position in den wiedergegebenen Sounddaten anzeigt.

Wenn eine Anwendung die `Sound.play()`-Methode aufruft, wird eine neue Instanz der SoundChannel-Klasse erstellt, mit der die Wiedergabe gesteuert wird.

Ihre Anwendung kann einen Sound ab einer bestimmten Startposition wiedergeben, indem diese Position als Wert in Millisekunden an den `startTime`-Parameter der `Sound.play()`-Methode übergeben wird. Sie kann auch einen Wert angeben, wie oft der Sound kurz hintereinander wiederholt wird, indem ein numerischer Wert im `loops`-Parameter der `Sound.play()`-Methode übergeben wird.

Wird die `Sound.play()`-Methode mit einem `startTime`-Parameter und einem `loops`-Parameter übergeben, wird der Sound mehrmals hintereinander ab dem gleichen Startpunkt wiedergegeben. Betrachten Sie dazu das folgende Codebeispiel:

```
var snd:Sound = new Sound(new URLRequest("repeatingSound.mp3"));
snd.play(1000, 3);
```

In diesem Beispiel wird der Sound drei Mal hintereinander ab einem Punkt eine Sekunde nach dem Start des Sounds wiedergegeben.

Unterbrechen und Fortsetzen eines Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Ihre Anwendung längere Sounds wiedergibt (wie Musikstücke oder Podcasts), sollen die Benutzern die Möglichkeit haben, die Wiedergabe dieser Sounds zu unterbrechen und fortzusetzen. Ein Sound kann während der Wiedergabe in ActionScript nicht wirklich unterbrochen, sondern nur angehalten werden. Die Wiedergabe eines Sounds kann jedoch an jedem beliebigen Punkt beginnen. Sie können die Position im Sound zum Zeitpunkt des Stoppens aufzeichnen und dann später die Wiedergabe an dieser Position fortsetzen.

Angenommen, Ihr Code lädt eine Sounddatei wie die folgende und gibt sie wieder:

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var channel:SoundChannel = snd.play();
```

Während der Sound wiedergegeben wird, kennzeichnet die `SoundChannel.position`-Eigenschaft die Stelle der Sounddatei, die gerade wiedergegeben wird. Ihre Anwendung kann den Positionswert speichern, bevor die Soundwiedergabe gestoppt wird. Dies ist z. B. mit dem folgenden Code möglich:

```
var pausePosition:int = channel.position;
channel.stop();
```

Um die Soundwiedergabe fortzusetzen, übergeben Sie den zuvor gespeicherten Positionswert, um die Soundwiedergabe am gleichen Punkt zu starten, an dem sie zuvor gestoppt wurde.

```
channel = snd.play(pausePosition);
```

Überwachen der Wiedergabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ihrer Anwendung muss wissen, wann ein Sound beendet ist, sodass ein weiterer Sound wiedergegeben oder bestimmte Ressourcen freigegeben werden können, die für die abgeschlossene Wiedergabe verwendet wurden. Die SoundChannel-Klasse löst ein `Event.SOUND_COMPLETE`-Ereignis aus, nachdem ein Sound vollständig wiedergegeben wurde. Ihre Anwendung kann auf dieses Ereignis überwachen und eine entsprechende Aktion ausführen. Betrachten Sie dazu den folgenden Beispielcode:

```
import flash.events.Event;
import flash.media.Sound;
import flash.net.URLRequest;

var snd:Sound = new Sound();
var req:URLRequest = new URLRequest("smallSound.mp3");
snd.load(req);

var channel:SoundChannel = snd.play();
channel.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

public function onPlaybackComplete(event:Event)
{
    trace("The sound has finished playing.");
}
```

Die SoundChannel-Klasse löst während der Wiedergabe keine Fortschrittereignisse aus. Um den Fortschritt der Wiedergabe zu melden, muss Ihre Anwendung einen eigenen Timermechanismus einrichten und die Position des Sound-Abspielkopfs verfolgen.

Zur Berechnung des Prozentsatzes, der von einem Sound wiedergegeben wurde, können Sie den Wert der `SoundChannel.position`-Eigenschaft durch die Länge der Sounddaten dividieren, die bereits wiedergegeben wurden:

```
var playbackPercent:uint = 100 * (channel.position / snd.length);
```

Jedoch meldet dieser Code nur dann den genauen Prozentsatz der wiedergegebenen Sounddaten, wenn die Datei vollständig geladen war, bevor die Wiedergabe gestartet wurde. Die `Sound.length`-Eigenschaft zeigt die Größe der aktuell geladenen Sounddaten, nicht die tatsächliche Größe der gesamten Sounddatei an. Um den Fortschritt bei der Wiedergabe eines Streaming-Sounds zu verfolgen, der noch geladen wird, muss Ihre Anwendung die tatsächliche Größe der gesamten Sounddatei schätzen und diesen Wert in den Berechnungen verwenden. Sie können die tatsächliche Länge der Sounddaten mithilfe der `bytesLoaded`- und `bytesTotal`-Eigenschaften des Sound-Objekts schätzen. Dazu wird z. B. der folgende Code verwendet:

```
var estimatedLength:int =
    Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
var playbackPercent:uint = 100 * (channel.position / estimatedLength);
```

Mit dem folgenden Code wird eine größere Sounddatei geladen und als Timermechanismus das `Event.ENTER_FRAME`-Ereignis verwendet, um den Fortschritt bei der Wiedergabe anzuzeigen. Der Code meldet regelmäßig den Prozentsatz der wiedergegebenen Sounddaten, der aus dem Wert für die aktuelle Position dividiert durch die Gesamtlänge der Sounddaten berechnet wird:

```
import flash.events.Event;
import flash.media.Sound;
import flash.net.URLRequest;

var snd:Sound = new Sound();
var req:URLRequest = new
    URLRequest("http://av.adobe.com/podcast/csbu_dev_podcast_epi_2.mp3");
snd.load(req);

var channel:SoundChannel;
channel = snd.play();
addEventListener(Event.ENTER_FRAME, onEnterFrame);
channel.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

function onEnterFrame(event:Event):void
{
    var estimatedLength:int =
        Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
    var playbackPercent:uint =
        Math.round(100 * (channel.position / estimatedLength));
    trace("Sound playback is " + playbackPercent + "% complete.");
}

function onPlaybackComplete(event:Event)
{
    trace("The sound has finished playing.");
    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

Nachdem die Wiedergabe der Sounddaten gestartet wurde, ruft der Code die `snd.play()`-Methode auf und speichert das resultierende SoundChannel-Objekt in der Variablen `channel`. Dann fügt er der Hauptanwendung einen Ereignis-Listener für das `Event.ENTER_FRAME`-Ereignis und dem SoundChannel-Objekt einen weiteren Ereignis-Listener für das `Event.SOUND_COMPLETE`-Ereignis hinzu, das auftritt, wenn die Wiedergabe abgeschlossen ist.

Jedes Mal, wenn in die Anwendung ein neues Bild in der Animation erreicht, wird die `onEnterFrame()`-Methode aufgerufen. Die `onEnterFrame()`-Methode schätzt die Gesamtlänge der Sounddatei basierend auf der bereits geladenen Datenmenge, berechnet dann die abgeschlossene Wiedergabe und zeigt diese an.

Wenn der gesamte Sound wiedergegeben wurde, werden die `onPlaybackComplete()`-Methode ausgeführt und der Ereignis-Listener für das `Event.ENTER_FRAME`-Ereignis gelöscht, sodass es nicht versucht, den Fortschritt weiter zu aktualisieren, obwohl die Wiedergabe abgeschlossen ist.

Das `Event.ENTER_FRAME`-Ereignis kann mehrmals pro Sekunde ausgelöst werden. In einigen Fällen soll der Wiedergabefortschritt weniger häufig angezeigt werden. Dann kann die Anwendung mithilfe der flash.util.Timer-Klasse einen eigenen Timermechanismus einrichten. Weitere Informationen finden Sie unter „Arbeiten mit Datum und Zeit“ auf Seite 1.

Anhalten von Streaming-Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Anzeige des Fortschritts bei der Wiedergabe von Streaming-Sounds (d. h. Sounds, die noch geladen werden, obwohl die Wiedergabe bereits begonnen hat), stellt einen Sonderfall dar. Wenn Ihre Anwendung die `SoundChannel.stop()`-Methode einer SoundChannel-Instanz aufruft, die einen Streaming-Sound wiedergibt, stoppt die Soundwiedergabe für ein Bild und startet die Soundwiedergabe dann im nächsten Bild von vorne. Dies tritt auf, da der Sound noch immer geladen wird. Um sowohl das Laden als auch die Wiedergabe eines Streaming-Sounds zu stoppen, rufen Sie die `Sound.close()`-Methode auf.

Sicherheitsüberlegungen beim Laden und Wiedergeben von Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Fähigkeit Ihrer Anwendung, auf Sounddaten zuzugreifen, kann auf das Flash Player- oder AIR-Sicherheitsmodell eingeschränkt werden. Jeder Sound unterliegt den Einschränkungen zweier verschiedener Sicherheits-Sandboxen, der Sandbox für den Inhalt selbst (die Inhalts-Sandbox) und die Sandbox für die Anwendung oder das Objekt, das den Sound lädt und wiedergibt (die Eigentümer-Sandbox). Im Fall von AIR-Anwendungsinhalt, der in der Sicherheits-Sandbox der Anwendung ausgeführt wird, sind sämtliche Sounds, einschließlich der aus anderen Domänen geladenen, für Inhalt in der Sicherheits-Sandbox der Anwendung zugänglich. Für Inhalte in Sicherheits-Sandboxen von anderen Anwendungen gelten jedoch die gleichen Regeln wie für Inhalt, der in Flash Player ausgeführt wird. Weitere allgemeine Informationen zum Sicherheitsmodell von Flash Player und zur Definition einer Sandbox finden Sie unter „Sicherheit“ auf Seite 1115.

Die Inhalts-Sandbox bestimmt, welche Sounddaten mit der `id3`-Eigenschaft oder der `SoundMixer.computeSpectrum()`-Methode aus dem Sound extrahiert werden können. Sie schränkt nicht das Laden oder die Wiedergabe der Sounddatei selbst ein.

Die Ursprungsdomäne der Sounddatei bestimmt die Sicherheitseinschränkungen der Inhalts-Sandbox. Allgemein gilt, wenn sich eine Sounddatei in der gleichen Domäne oder im gleichen Ordner wie die SWF-Datei der Anwendung oder dem Objekt befindet, die bzw. das den Sound lädt, hat die Anwendung bzw. das Objekt vollständigen Zugriff auf die Sounddatei. Stammt der Sound aus einer anderen Domäne als die Anwendung, kann er mit einer Richtliniendatei dennoch in die Inhalts-Sandbox gebracht werden.

Ihre Anwendung kann ein SoundLoaderContext-Objekt mit einer `checkPolicyFile`-Eigenschaft als Parameter an die `Sound.load()`-Methode übergeben. Durch Einstellen der `checkPolicyFile`-Eigenschaft auf `true` weisen Sie Flash Player bzw. AIR an, auf dem Server, von dem der Sound geladen wird, nach einer Richtliniendatei zu suchen. Wenn eine Richtliniendatei vorhanden ist und sie den Zugriff auf die Domäne der ladenden SWF-Datei erteilt, kann die SWF-Datei die Sounddatei laden, auf die `id3`-Eigenschaft des Sound-Objekts zugreifen und die `SoundMixer.computeSpectrum()`-Methode für geladene Sounds aufrufen.

Die Eigentümer-Sandbox steuert die lokale Soundwiedergabe. Die Eigentümer-Sandbox wird durch die Anwendung oder das Objekt definiert, die bzw. das die Wiedergabe eines Sounds startet.

Solange die folgenden Kriterien erfüllt sind, schaltet die `SoundMixer.stopAll()`-Methode alle Sounds in allen aktuell wiedergegebenen SoundChannel-Objekten stumm:

- Die Sounds wurden von Objekten innerhalb der gleichen Eigentümer-Sandbox gestartet.

- Die Sounds stammen von einer Quelle mit einer Richtliniendatei, die den Zugriff auf die Domäne der Anwendung oder des Objekts gestattet, die bzw. das die `SoundMixer.stopAll()`-Methode aufruft.

Jedoch unterliegt in einer AIR-Anwendung Inhalt, der in der Sicherheits-Sandbox der Anwendung ausgeführt wird (Inhalt, der mit der AIR-Anwendung installiert wird), nicht diesen Sicherheitseinschränkungen.

Um festzustellen, ob die `SoundMixer.stopAll()`-Methode tatsächlich die Wiedergabe aller Sounds stoppt, kann Ihre Anwendung die `SoundMixer.areSoundsInaccessible()`-Methode aufrufen. Wenn diese Methode den Wert `true` zurückgibt, befinden sich einige der wiedergegebenen Sounds außerhalb der Kontrolle der aktuellen Eigentümer-Sandbox und können somit nicht von der `SoundMixer.stopAll()`-Methode gestoppt werden.

Die `SoundMixer.stopAll()`-Methode stoppt darüber hinaus den Abspielkopf bei allen Sounds, die aus externen Dateien geladen wurden. Die Wiedergabe von Sounds, die in FLA-Dateien eingebettet sind und mithilfe des Flash-Authoring-Tools an Bilder in der Zeitleiste angefügt wurden, wird möglicherweise neu gestartet, wenn die Animation zu einem neuen Bild wechselt.

Steuern der Lautstärke und Richtungseinstellung des Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein einzelnes SoundChannel-Objekt steuert den linken und rechten Stereokanal eines Sounds. Wenn ein MP3-Sound nur mono vorliegt, weisen der linke und rechte Kanal des SoundChannel-Objekts identische Wellenformen auf.

Sie können die Amplitude jedes Stereokanals eines wiedergegebenen Sounds leicht mithilfe der Eigenschaften `leftPeak` und `rightPeak` des SoundChannel-Objekts ermitteln. Diese Eigenschaften zeigen die Amplitudenspitze der Soundwellenform selbst an. Sie zeigen nicht die tatsächliche Wiedergabelautstärke an. Die tatsächliche Wiedergabelautstärke ist eine Funktion der Soundwellenamplitude und der Lautstärkewerte, die im SoundChannel-Objekt und der SoundMixer-Klasse eingestellt wurden.

Die pan-Eigenschaft eines SoundChannel-Objekts kann dazu verwendet werden, während der Wiedergabe eine unterschiedliche Lautstärke für den rechten und linken Kanal festzulegen. Die pan-Eigenschaft kann einen Wert zwischen -1 und 1 annehmen. -1 steht dabei für volle Lautstärke auf dem linken Kanal und Stille auf dem rechten, während der Wert 1 volle Lautstärke auf dem rechten Kanal und Stille auf dem linken Kanal bedeutet. Numerische Werte zwischen -1 und 1 stellen proportionale Werte für den linken und rechten Kanal ein. Ein Wert von 0 bedeutet, dass beide Kanäle mit der gleichen Lautstärke wiedergegeben werden.

Mit dem folgenden Code wird ein SoundTransform-Objekt mit dem Lautstärkewert 0,6 und dem Wert -1 für die Richtungseinstellung erstellt (linker Kanal mit voller Lautstärke und Stille auf dem rechten Kanal). Der Code übergibt das SoundTransform-Objekt als Parameter an die `play()`-Methode, die das SoundTransform-Objekt auf das neue SoundChannel-Objekt anwendet, das zur Wiedergabesteuerung erstellt wurde.

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var trans:SoundTransform = new SoundTransform(0.6, -1);
var channel:SoundChannel = snd.play(0, 1, trans);
```

Sie können Lautstärke und Richtungseinstellung während der Soundwiedergabe ändern, indem Sie die Eigenschaften `pan` oder `volume` eines SoundTransform-Objekts festlegen und dann das Objekt als `soundTransform`-Eigenschaft eines SoundChannel-Objekts anwenden.

Mit der `soundTransform`-Eigenschaft der SoundMixer-Klasse können Sie die Werte für die globale Lautstärke und die Richtungseinstellung auch für alle Sounds gleichzeitig einstellen. Dies wird im folgenden Codebeispiel gezeigt:

```
SoundMixer.soundTransform = new SoundTransform(1, -1);
```

Sie können auch ein SoundTransform-Objekt verwenden, um Lautstärke und Richtungseinstellung für ein Microphone-Objekt (siehe „Erfassen von Soundeingaben“ auf Seite 492) und für Sprite- und SimpleButton-Objekte festzulegen.

Mit dem folgenden Beispielcode wird die Richtungseinstellung des Sounds während der Wiedergabe vom linken zum rechten Kanal und zurück gewechselt.

```
import flash.events.Event;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundMixer;
import flash.net.URLRequest;

var snd:Sound = new Sound();
var req:URLRequest = new URLRequest("bigSound.mp3");
snd.load(req);

var panCounter:Number = 0;

var trans:SoundTransform;
trans = new SoundTransform(1, 0);
var channel:SoundChannel = snd.play(0, 1, trans);
channel.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

addEventListener(Event.ENTER_FRAME, onEnterFrame);

function onEnterFrame(event:Event):void
{
    trans.pan = Math.sin(panCounter);
    channel.soundTransform = trans; // or SoundMixer.soundTransform = trans;
    panCounter += 0.05;
}

function onPlaybackComplete(event:Event):void
{
    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

Der Code beginnt mit dem Laden einer Sounddatei und erstellt dann ein neues SoundTransform-Objekt mit der Lautstärkeeinstellung 1 (volle Lautstärke) und der Richtungseinstellung 0 (gleichmäßige Aufteilung zwischen dem linken und rechten Kanal). Dann ruft der Code die `snd.play()`-Methode auf und übergibt das SoundTransform-Objekt als Parameter.

Während der Soundwiedergabe wird die `onEnterFrame()`-Methode wiederholt ausgeführt. Die `onEnterFrame()`-Methode verwendet die `Math.sin()`-Funktion zum Generieren eines Wertes zwischen -1 und 1. Dieser Bereich entspricht den gültigen Werten der `SoundTransform.pan`-Eigenschaft. Die `pan`-Eigenschaft des SoundTransform-Objekts wird auf den neuen Wert eingestellt, dann wird die `soundTransform`-Eigenschaft des Kanals so eingestellt, dass sie das geänderte SoundTransform-Objekt verwendet.

Um dieses Beispiel auszuführen, ersetzen Sie den Dateinamen „bigSound.mp3“ durch den Namen einer lokalen MP3-Datei. Dann führen Sie das Beispiel aus: Sie sollten hören, wie der Lautstärke des linken Kanals ansteigt, während die Lautstärke des rechten Kanals abnimmt, und umgekehrt.

In diesem Beispiel kann der gleiche Effekt durch Einstellen der `soundTransform`-Eigenschaft der SoundMixer-Klasse erreicht werden. Dies würde sich jedoch auf die Richtungseinstellung aller aktuell wiedergegebenen Sounds auswirken, und nicht nur auf den Sound, der von diesem SoundChannel-Objekt wiedergegeben wird.

Verwenden von Sound-Metadaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sounddateien im MP3-Format können zusätzliche Daten zum Sound in Form von ID3-Tags enthalten.

Nicht jede MP3-Datei enthält ID3-Metadaten. Wenn ein Sound-Objekt eine MP3-Sounddatei lädt, wird ein `Event.ID3`-Ereignis ausgelöst, falls die Sounddatei ID3-Metadaten enthält. Um Laufzeitfehler zu verhindern, sollte die Anwendung warten, bis sie ein `Event.ID3`-Ereignis empfangen hat. Erst dann sollte sie versuchen, auf die `Sound.id3`-Eigenschaft eines geladenen Sounds zuzugreifen.

Der folgende Code zeigt, wie die ID3-Metadaten einer geladenen Sounddatei erkannt werden:

```
import flash.events.Event;
import flash.media.ID3Info;
import flash.media.Sound;

var s:Sound = new Sound();
s.addEventListener(Event.ID3, onID3InfoReceived);
s.load("mySound.mp3");

function onID3InfoReceived(event:Event)
{
    var id3:ID3Info = event.target.id3;

    trace("Received ID3 Info:");
    for (var propName:String in id3)
    {
        trace(propName + " = " + id3[propName]);
    }
}
```

Dieser Code beginnt mit dem Erstellen eines Sound-Objekts, das anschließend das `Event.ID3`-Ereignis überwachen soll. Wenn die ID3-Metadaten der Sounddatei geladen wurden, wird die `onID3InfoReceived()`-Methode aufgerufen. Das Ziel des Event-Objekts, das an die `onID3InfoReceived()`-Methode übergeben wird, ist das ursprüngliche Sound-Objekt. So erhält die Methode die `id3`-Eigenschaft des Sound-Objekts und durchläuft dann alle benannten Eigenschaften, um deren Werte abzurufen.

Zugreifen auf Raw-Sounddaten

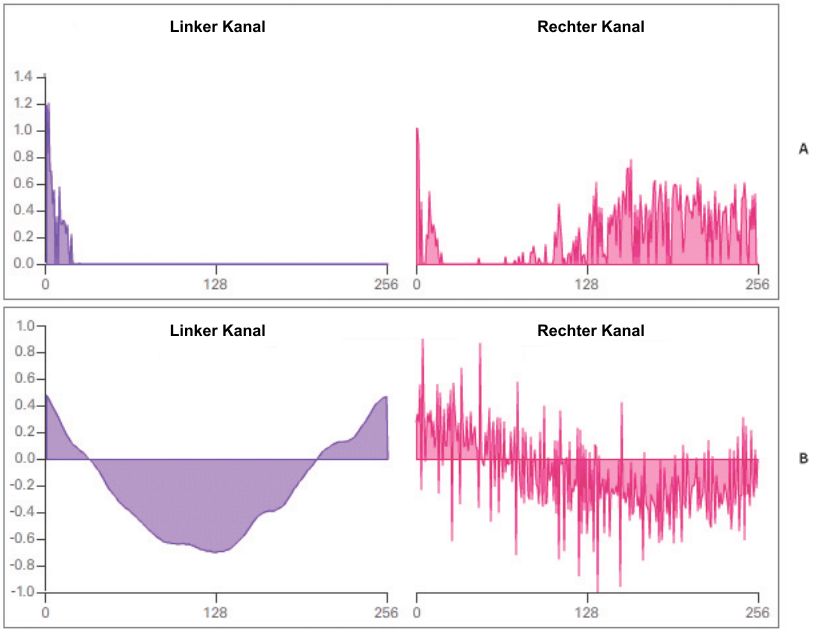
Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `SoundMixer.computeSpectrum()`-Methode kann eine Anwendung die Raw-Sounddaten der aktuell wiedergegebenen Wellenform einlesen. Wenn aktuell mehrere SoundChannel-Objekte wiedergegeben werden, zeigt die `SoundMixer.computeSpectrum()`-Methode die kombinierten und gemischten Sounddaten aller SoundChannel-Objekte an.

Die Sounddaten werden als ein ByteArray-Objekt mit 512 Byte Dateninhalt zurückgegeben. Jedes Objekt enthält eine Gleitkommazahl zwischen -1 und 1. Diese Werte repräsentieren die Amplitude der Punkte in der wiedergegebenen Soundwellenform. Die Werte werden in zwei 256er-Gruppen bereitgestellt, die erste Gruppe für den linken Stereokanal und die zweite Gruppe für den rechten Stereokanal.

Wenn der `FFTMode`-Parameter auf `true` gesetzt ist, gibt die `SoundMixer.computeSpectrum()`-Methode anstelle von Wellenformdaten Frequenzspektrumdaten zurück. Das Frequenzspektrum zeigt eine nach der Soundfrequenz angeordnete Amplitude an, von der niedrigsten Frequenz zu höchsten. Zum Konvertieren der Wellenformdaten in Frequenzspektrumdaten wird ein Fast Fourier Transform (FFT) eingesetzt. Die resultierenden Frequenzspektrumwerte befinden sich im Bereich von 0 bis etwa 1,414 (Quadratwurzel aus 2).

Im folgenden Diagramm werden die von der `computeSpectrum()`-Methode zurückgegebenen Daten bei einer `FFTMode`-Parametereinstellung von `true` und einer Einstellung von `false` verglichen. Der Sound, dessen Daten in diesem Diagramm verwendet wurden, enthält einen lauten Basston auf dem linken Kanal und einen Schlagzeugton auf dem rechten Kanal.



Von der SoundMixer.computeSpectrum()-Methode zurückgegebene Werte
***A.** fftMode=true **B.** fftMode=false*

Die `computeSpectrum()`-Methode kann auch Daten zurückgeben, die mit einer niedrigeren Bitrate neu gesampelt wurden. Im Allgemeinen führt dies zu sanfteren Wellenformdaten oder Frequenzdaten bei geringeren Details. Der `stretchFactor`-Parameter bestimmt die Rate, mit der die `computeSpectrum()`-Methode Daten abtastet. Wenn der `stretchFactor`-Parameter auf 0 (die Standardeinstellung) eingestellt ist, werden die Sounddaten mit einer Rate von 44,1 kHz abgetastet. Die Abtastrate wird bei jedem nachfolgenden Wert für den stretchFactor-Parameter halbiert. Somit gibt der Wert 1 eine Abtastrate von 22,05 kHz an, der Wert 2 eine Abtastrate von 11,025 kHz usw. Die `computeSpectrum()`-Methode gibt weiterhin 256 Byte pro Stereokanal zurück, auch wenn ein höherer `stretchFactor`-Wert verwendet wird.

Die `SoundMixer.computeSpectrum()`-Methode weist einige Einschränkungen auf:

- Da Sounddaten von einem Mikrofon oder von RTMP-Streams das globale SoundMixer-Objekt nicht passieren, kann die `SoundMixer.computeSpectrum()`-Methode von diesen Quellen keine Daten zurückgeben.

- Stammt einer oder mehrere der wiedergegebenen Sounds von Quellen außerhalb der aktuellen Inhalts-Sandbox, wird aufgrund der Sicherheitseinstellungen ein Fehler durch die `SoundMixer.computeSpectrum()`-Methode auslöst. Nähere Einzelheiten über die Sicherheitseinschränkungen der `SoundMixer.computeSpectrum()`-Methode finden Sie unter „Sicherheitsüberlegungen beim Laden und Wiedergeben von Sounds“ auf Seite 485 und „Zugriff auf geladene Medien als Daten“ auf Seite 1142.

Jedoch unterliegt in einer AIR-Anwendung Inhalt, der in der Sicherheits-Sandbox der Anwendung ausgeführt wird (Inhalt, der mit der AIR-Anwendung installiert wird), nicht diesen Sicherheitseinschränkungen.

Erstellen eines einfachen Anzeigeprogramms für Sounds

Flash Player 9 und höher, Adobe AIR 1.0 und höher

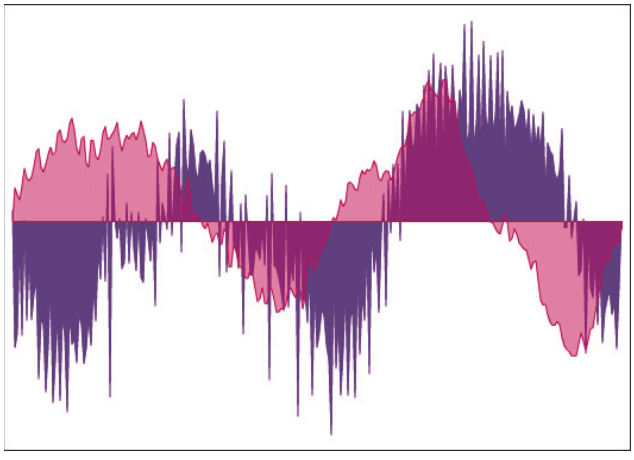
Im folgenden Beispiel wird die `SoundMixer.computeSpectrum()`-Methode verwendet, um ein Diagramm der Sound-Wellenform anzuzeigen, die mit jedem Bild animiert wird:

```
import flash.display.Graphics;
import flash.events.Event;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundMixer;
import flash.net.URLRequest;

const PLOT_HEIGHT:int = 200;
const CHANNEL_LENGTH:int = 256;

var snd:Sound = new Sound();
var req:URLRequest = new URLRequest("bigSound.mp3");
snd.load(req);

var channel:SoundChannel;
channel = snd.play();
addEventListener(Event.ENTER_FRAME, onEnterFrame);
snd.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

var bytes:ByteArray = new ByteArray();

function onEnterFrame(event:Event):void
{
    SoundMixer.computeSpectrum(bytes, false, 0);

    var g:Graphics = this.graphics;

    g.clear();
    g.lineStyle(0, 0x6600CC);
    g.beginFill(0x6600CC);
    g.moveTo(0, PLOT_HEIGHT);

    var n:Number = 0;

    // left channel
    for (var i:int = 0; i < CHANNEL_LENGTH; i++)
    {
        n = (bytes.readFloat() * PLOT_HEIGHT);
        g.lineTo(i * 2, PLOT_HEIGHT - n);
    }
```

```
    g.lineTo(CHANNEL_LENGTH * 2, PLOT_HEIGHT);
    g.endFill();

    // right channel
    g.lineStyle(0, 0xCC0066);
    g.beginFill(0xCC0066, 0.5);
    g.moveTo(CHANNEL_LENGTH * 2, PLOT_HEIGHT);

    for (i = CHANNEL_LENGTH; i > 0; i--)
    {
        n = (bytes.readFloat() * PLOT_HEIGHT);
        g.lineTo(i * 2, PLOT_HEIGHT - n);
    }
    g.lineTo(0, PLOT_HEIGHT);
    g.endFill();
}

function onPlaybackComplete(event:Event)
{
    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

In diesem Beispiel wird zunächst eine Sounddatei geladen und wiedergegeben und dann das `Event.ENTER_FRAME`-Ereignis überwacht, das während der Soundwiedergabe die `onEnterFrame()`-Methode auslöst. Die `onEnterFrame()`-Methode beginnt mit dem Aufruf der `SoundMixer.computeSpectrum()`-Methode, die die Soundwellendaten im ByteArray-Objekt `bytes` speichert.

Die Soundwellenform wird mithilfe der Vektorzeichnungs-API geplottet. Eine `for`-Schleife durchläuft die ersten 256 Datenwerten, die den linken Kanal darstellen, und zeichnet mithilfe der `Graphics.lineTo()`-Methode eine Linie von einem Punkt zum nächsten. Eine zweite `for`-Schleife durchläuft die nächsten 256 Werte und zeichnet sie diesmal in umgekehrter Reihenfolge, von rechts nach links. Die resultierenden Wellenformplots können einen interessanten Spiegeleffekt erzeugen. Dies wird im folgenden Bild gezeigt.



Erfassen von Soundeingaben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Microphone-Klasse kann Ihre Anwendung eine Verbindung zu einem Mikrofon oder einem anderen Sound-Eingabegerät des Benutzersystems herstellen und die eingegangenen Audiosignale an die Systemlautsprecher übertragen oder an einen Remote-Server, z. B. einen Flash Media Server, senden. Sie können auf die Raw-Audiodaten vom Mikrofon zugreifen und sie aufnehmen oder verarbeiten. Außerdem können Sie die Audiodaten direkt an die Lautsprecher des Systems senden oder komprimierte Audiodaten an einen Remote-Server senden. Für das Senden der Daten an einen Remote-Server können Sie den Speex- oder Nellymoser-Codec verwenden. (Der Speex-Codec wird ab Flash Player 10 und Adobe AIR 1.5 unterstützt.)

Verwandte Hilfethemen

Michael Chaize: AIR, Android, and the Microphone

Christophe Coenraets: Voice Notes for Android

Zugriff auf ein Mikrofon

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Microphone-Klasse hat keine Konstruktor-Methode. Stattdessen verwenden Sie die statische `Microphone.getMicrophone()`-Methode, um eine neue Microphone-Instanz zu erstellen, wie im Folgenden dargestellt:

```
var mic:Microphone = Microphone.getMicrophone();
```

Das Aufrufen der `Microphone.getMicrophone()`-Methode ohne einen Parameter gibt das erste auf dem Benutzersystem erfasste Sound-Eingabegerät zurück.

Es können mehrere Sound-Eingabegeräte an ein System angeschlossen sein. Ihre Anwendung kann mithilfe der `Microphone.names`-Eigenschaft ein Array mit den Namen aller verfügbaren Sound-Eingabegeräte erstellen. Dann ruft sie die `Microphone.getMicrophone()`-Methode mit dem `index`-Parameter auf, der dem Indexpositionswert eines Gerätenamens im Array entspricht.

Es kann auch sein, dass weder ein Mikrofon noch ein anderes Sound-Eingabegerät an das System angeschlossen ist. Mit der `Microphone.names`-Eigenschaft oder der `Microphone.getMicrophone()`-Methode können Sie prüfen, ob im System des Benutzers ein Sound-Eingabegerät installiert ist. Wenn der Benutzer kein Sound-Eingabegerät installiert hat, weist das `names`-Array eine Länge von Null auf und die `getMicrophone()`-Methode gibt den Wert `null` zurück.

Wenn Ihre Anwendung die `Microphone.getMicrophone()`-Methode aufruft, zeigt Flash Player das Dialogfeld „Flash Player-Einstellungen“ an. In diesem Dialogfeld wird der Benutzer aufgefordert, Flash Player den Zugriff auf an das System angeschlossene Sound-Eingabegeräte entweder zu gestatten oder zu verweigern. Nachdem der Benutzer auf die Schaltfläche „Zulassen“ oder „Verweigern“ geklickt hat, wird ein StatusEvent ausgelöst. Die `code`-Eigenschaft dieser StatusEvent-Instanz gibt an, ob der Zugriff auf das Mikrofon zugelassen oder verweigert wurde. Dies wird im Folgenden gezeigt:

```
import flash.media.Microphone;

var mic:Microphone = Microphone.getMicrophone();
mic.addEventListener(StatusEvent.STATUS, this.onMicStatus);

function onMicStatus(event:StatusEvent):void
{
    if (event.code == "Microphone.Unmuted")
    {
        trace("Microphone access was allowed.");
    }
    else if (event.code == "Microphone.Muted")
    {
        trace("Microphone access was denied.");
    }
}
```

Die `StatusEvent.code`-Eigenschaft enthält „Microphone.Unmuted“, wenn der Zugriff zugelassen wurde, oder „Microphone.Muted“, wenn der Zugriff verweigert wurde.

Die `Microphone.muted`-Eigenschaft ist auf `true` eingestellt, wenn der Benutzer den Zugriff auf das Mikrofon gestattet, oder auf `false`, wenn der Zugriff verweigert wurde. Die `muted`-Eigenschaft der Microphone-Instanz wird jedoch erst nach dem Auslösen von StatusEvent eingestellt. Ihre Anwendung muss also auch auf das `StatusEvent.STATUS`-Ereignis warten, bevor sie die `Microphone.muted`-Eigenschaft überprüft.

Damit Flash Player das Einstellungsdialogfeld anzeigt, muss das Anwendungsfenster groß genug sein (mindestens 215 x 138 Pixel). Andernfalls wird der Zugriff automatisch verweigert.

Inhalt, der in der AIR-Anwendungs-Sandbox ausgeführt wird, benötigt keine Erlaubnis des Benutzers für den Zugriff auf das Mikrofon. Deshalb werden nie Statusereignisse ausgelöst, wenn der Ton des Mikrofons ein- oder ausgeschaltet wird. Inhalt, der in AIR außerhalb der Anwendungs-Sandbox ausgeführt wird, benötigt die Erlaubnis des Benutzers, sodass diese Statusereignisse ausgelöst werden können.

Leiten des Audioeingangs von einem Mikrofon an die lokalen Lautsprecher

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Durch Aufrufen der `Microphone.setLoopback()`-Methode mit dem Parameterwert `true` kann der Audioeingang von einem Mikrofon an das lokale Lautsprechersystem geleitet werden.

Wenn Sound von einem lokalen Mikrofon an lokale Lautsprecher geleitet wird, besteht das Risiko einer Audio-Feedbackschleife (Rückkopplung), die ein lautes Geräusch verursacht und die Sound-Hardware potenziell beschädigen kann. Durch das Aufrufen der `Microphone.setUseEchoSuppression()`-Methode mit dem Parameter `true` wird das Risiko einer Rückkopplung reduziert, jedoch nicht vollständig beseitigt. Es wird empfohlen, stets `Microphone.setUseEchoSuppression(true)` aufzurufen, bevor `Microphone.setLoopback(true)` aufgerufen wird, es sei denn, Sie sind sicher, dass der Benutzer den Sound über einen Kopfhörer oder ein anderes Wiedergabesystem außer Lautsprechern wiedergibt.

Im folgenden Code wird gezeigt, wie Audio von einem lokalen Mikrofon an die lokalen Systemlautsprecher geleitet wird:

```
var mic:Microphone = Microphone.getMicrophone();
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
```

Ändern von Mikrofonaudiodaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ihre Anwendung kann die von einem Mikrofon aufgenommenen Audiodaten auf zwei Arten ändern. Zunächst kann sie die Signalstärke des Eingabesounds ändern, wodurch letztlich die Eingabewerte mit einem bestimmten Wert multipliziert werden, um einen lauteren oder leiseren Sound zu erzeugen. Die `Microphone.gain`-Eigenschaft akzeptiert numerische Werte zwischen 0 und 100 (einschließlich). Ein Wert von 50 verhält sich wie ein Multiplikator von eins und steht für eine normale Lautstärke. Ein Wert von Null verhält sich wie ein Multiplikator von Null und schaltet das Eingangsaudio stumm. Werte oberhalb von 50 stehen für eine höhere als die normale Lautstärke.

Außerdem kann Ihre Anwendung die Abtastrate des Eingangsaudios ändern. Höhere Abtastraten ergeben eine bessere Soundqualität, erzeugen jedoch auch dichtere Datenströme, die bei der Übertragung und Speicherung mehr Ressourcen belegen. Die `Microphone.rate`-Eigenschaft steht für die Audio-Abtastrate, die in Kilohertz (kHz) gemessen wird. Die Standard-Abtastrate beträgt 8 kHz. Sie können die `Microphone.rate`-Eigenschaft auf einen Wert höher als 8 kHz einstellen, wenn Ihr Mikrofon die höhere Abtastrate unterstützt. Das Setzen der `Microphone.rate`-Eigenschaft auf den Wert 11 stellt die Abtastrate auf 11 kHz ein. Das Einstellen auf den Wert 22 führt zu einer Abtastrate von 22 kHz usw. Welche Abtastraten zur Verfügung stehen, hängt vom ausgewählten Codec ab. Wenn Sie den Nellymoser-Codec verwenden, können Sie als Abtastrate 5, 8, 11, 16, 22 und 44 kHz festlegen. Wenn Sie den Speex-Codec verwenden (verfügbar ab Flash Player 10 und Adobe AIR 1.5), können Sie nur 16 kHz verwenden.

Erfassen einer Mikrofonaktivität

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player versucht zu erkennen, wenn kein Sound über ein Mikrofon übertragen wird, um Bandbreite und Verarbeitungsressourcen zu schonen. Bleibt die Mikrofonaktivität über einen bestimmten Zeitraum unter der Ruhelautstärke, stoppt Flash Player die Übertragung des Audioeingangs und löst stattdessen ein einfaches ActivityEvent aus. Wenn Sie den Speex-Codec verwenden (verfügbar ab Flash Player 10 und Adobe AIR 1.5), stellen Sie die Ruhelautstärke auf 0, um sicherzustellen, dass die Anwendung kontinuierlich Audiodaten überträgt. Die Speex-Tonaktivitätserkennung verringert automatisch die Bandbreite.

Hinweis: *Ein Microphone-Objekt löst nur Aktivitätsereignisse aus, wenn die Anwendung das Mikrofon überwacht. Wenn Sie `setLoopBack( true )` nicht aufrufen, sollten Sie deshalb einen Listener für SampleData-Ereignisse hinzufügen oder das Mikrofon einem NetStream-Objekt zuordnen. Es werden dann keine Aktivitätsereignisse ausgelöst.*

Das Erfassen einer Mikrofonaktivität wird von drei Eigenschaften der Microphone-Klasse überwacht und gesteuert:

- Die schreibgeschützte `activityLevel`-Eigenschaft gibt den Betrag des von einem Mikrofon erfassten Sounds auf einer Skala zwischen 0 und 100 an.
- Die `silenceLevel`-Eigenschaft legt die zum Aktivieren des Mikrofons und zum Auslösen eines `ActivityEvent.ACTIVITY`-Ereignisses erforderliche Lautstärke fest. Auch die `silenceLevel`-Eigenschaft verwendet eine Skala von 0 bis 100; der Standardwert ist 10.
- Die `silenceTimeout`-Eigenschaft legt einen Zeitraum in Millisekunden fest, über den die Mikrofonaktivität unter der Ruhelautstärke bleiben muss, damit ein `ActivityEvent.ACTIVITY`-Ereignis ausgelöst wird. Ein solches Ereignis kennzeichnet, dass das Mikrofon jetzt stumm ist. Der Standardwert für `silenceTimeout` ist 2000.

Die Eigenschaften `Microphone.silenceLevel` und `Microphone.silenceTimeout` sind schreibgeschützt, ihre Werte können jedoch mithilfe der `Microphone.setSilenceLevel()`-Methode geändert werden.

In einigen Fällen kann das Aktivieren des Mikrofons eine kurze Verzögerung verursachen, wenn eine neue Aktivität erfasst wird. Diese Aktivierungsverzögerungen können vermieden werden, indem das Mikrofon jederzeit aktiv bleibt. Ihre Anwendung kann die `Microphone.setSilenceLevel()`-Methode mit einem Wert Null für den Parameter `silenceLevel` aufrufen, um Flash Player anzuweisen, das Mikrofon stets aktiv zu halten und Audiodaten auch dann aufzunehmen, wenn kein Sound erfasst wird. Entsprechend verhindert der Wert 100 für den Parameter `silenceLevel`, dass das Mikrofon überhaupt aktiviert wird.

Mit dem folgenden Beispielcode werden Informationen zum Mikrofon und Berichte zu Aktivitätsereignissen und Statusereignissen angezeigt, die von einem Microphone-Objekt ausgelöst werden:

```
import flash.events.ActivityEvent;
import flash.events.StatusEvent;
import flash.media.Microphone;

var deviceArray:Array = Microphone.names;
trace("Available sound input devices:");
for (var i:int = 0; i < deviceArray.length; i++)
{
    trace(" " + deviceArray[i]);
}

var mic:Microphone = Microphone.getMicrophone();
mic.gain = 60;
mic.rate = 11;
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
mic.setSilenceLevel(5, 1000);

mic.addEventListener(ActivityEvent.ACTIVITY, this.onMicActivity);
mic.addEventListener(StatusEvent.STATUS, this.onMicStatus);

var micDetails:String = "Sound input device name: " + mic.name + '\n';
micDetails += "Gain: " + mic.gain + '\n';
micDetails += "Rate: " + mic.rate + " kHz" + '\n';
micDetails += "Muted: " + mic.muted + '\n';
micDetails += "Silence level: " + mic.silenceLevel + '\n';
micDetails += "Silence timeout: " + mic.silenceTimeout + '\n';
micDetails += "Echo suppression: " + mic.useEchoSuppression + '\n';
trace(micDetails);

function onMicActivity(event:ActivityEvent):void
{
    trace("activating=" + event.activating + ", activityLevel=" +
        mic.activityLevel);
}

function onMicStatus(event:StatusEvent):void
{
    trace("status: level=" + event.level + ", code=" + event.code);
}
```

Wenn Sie das oben stehende Beispiel ausführen, sprechen Sie in das Systemmikrofon oder erzeugen Sie ein Geräusch, und achten Sie dabei auf die trace-Anweisungen, die im Konsolen- oder Debugging-Fenster angezeigt werden.

Senden von Audiodaten an einen Media Server oder Empfangen von Audiodaten von einem Media Server

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie ActionScript mit einem Streaming Media Server wie dem Flash Media Server verwenden, stehen Ihnen zusätzliche Audiofunktionen zur Verfügung.

Genauer gesagt, Ihre Anwendung kann ein Microphone-Objekt an ein NetStream-Objekt anhängen und Daten direkt vom Benutzermikrofon an den Server übertragen. Audiodaten können auch vom Server an eine Anwendung gestreamt und als Teil eines Movieclips oder mithilfe eines Video-Objekts wiedergegeben werden.

Der Speex-Codec ist ab Flash Player 10 und Adobe AIR 1.5 verfügbar. Um den Codec festzulegen, der für an den Medienserver gesendete Audiodaten verwendet wird, stellen Sie die `codec`-Eigenschaft des Microphone-Objekts ein. Diese Eigenschaft kann zwei Werte annehmen, die in der SoundCodec-Klasse festgelegt werden. Durch Setzen der codec-Eigenschaft auf `SoundCodec.SPEEX` wird der Speex-Codec für die Komprimierung von Audiodaten gewählt. Durch Setzen der Eigenschaft auf `SoundCodec.NELLYMOSER` (die Standardeinstellung), wird der Nellymoser-Codec für die Komprimierung von Audiodaten gewählt.

Weitere Informationen finden Sie in der Dokumentation zu Flash Media Server unter www.adobe.com/go/learn_fms_docs_de.

Erfassen von Sounddaten über ein Mikrofon

Flash Player 10.1 und höher, Adobe AIR 2 und höher

In Flash Player 10.1. und AIR 2 oder höher können Sie Sounddaten über ein Mikrofon als Byte-Array aus Gleitkommawerten erfassen. Jeder Wert entspricht einem Sample der monofonischen Audiodaten.

Zum Erfassen von Daten über ein Mikrofon legen Sie einen Ereignis-Listener für das`sampleData`-Ereignis des Microphone-Objekts fest. Das Microphone-Objekt löst regelmäßig `sampleData`-Ereignisse aus, während der Mikrofonpuffer mit Soundsamples gefüllt wird. Das SampleDataEvent-Objekt verfügt über eine `data`-Eigenschaft, bei der es sich um ein Byte-Array aus Soundsamples handelt. Die Samples werden als Gleitkommawerte dargestellt, die jeweils ein monofonisches Soundsample repräsentieren.

Mit dem folgenden Code werden über ein Mikrofon Sounddaten in einem ByteArray-Objekt namens `soundBytes` erfasst:

```
var mic:Microphone = Microphone.getMicrophone();
mic.setSilenceLevel(0, DELAY_LENGTH);
mic.addEventListener(SampleDataEvent.SAMPLE_DATA, micSampleDataHandler);
function micSampleDataHandler(event:SampleDataEvent):void {
    while(event.data.bytesAvailable) {
        var sample:Number = event.data.readFloat();
        soundBytes.writeFloat(sample);
    }
}
```

Sie können die Samplebyte zur Audiowiedergabe für ein Sound-Objekt wieder verwenden. In diesem Fall sollten Sie die `rate`-Eigenschaft des Microphone-Objekts auf 44 einstellen. Dies ist die von Sound-Objekten verwendete Abtastrate. (Sie können auch mit einer niedrigeren Abtastrate erfasste Mikrofonsamples in 44 kHz umwandeln, wie für das Sound-Objekt erforderlich.) Beachten Sie auch, dass das Microphone-Objekt monofonische Samples erfasst, während das Sound-Objekt Stereo verwendet. Deshalb müssen alle über das Microphone-Objekt erfassten Byte zweimal in das Sound-Objekt geschrieben werden. Im folgenden Beispiel werden Mikrofondaten mit einer Länge von 4 Sekunden erfasst und mithilfe eines Sound-Objekts wiedergegeben:

```
const DELAY_LENGTH:int = 4000;
var mic:Microphone = Microphone.getMicrophone();
mic.setSilenceLevel(0, DELAY_LENGTH);
mic.gain = 100;
mic.rate = 44;
mic.addEventListener(SampleDataEvent.SAMPLE_DATA, micSampleDataHandler);

var timer:Timer = new Timer(DELAY_LENGTH);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.start();

function micSampleDataHandler(event:SampleDataEvent):void
{
    while(event.data.bytesAvailable)
    {
        var sample:Number = event.data.readFloat();
        soundBytes.writeFloat(sample);
    }
}
var sound:Sound = new Sound();
var channel:SoundChannel;
function timerHandler(event:TimerEvent):void
{
    mic.removeEventListener(SampleDataEvent.SAMPLE_DATA, micSampleDataHandler);
    timer.stop();
    soundBytes.position = 0;
    sound.addEventListener(SampleDataEvent.SAMPLE_DATA, playbackSampleHandler);
    channel.addEventListener( Event.SOUND_COMPLETE, playbackComplete );
    channel = sound.play();
}

function playbackSampleHandler(event:SampleDataEvent):void
{
    for (var i:int = 0; i < 8192 && soundBytes.bytesAvailable > 0; i++)
    {
        trace(sample);
        var sample:Number = soundBytes.readFloat();
        event.data.writeFloat(sample);
        event.data.writeFloat(sample);
    }
}

function playbackComplete( event:Event ):void
{
    trace( "Playback finished.");
}
```

Weitere Informationen zum Wiedergeben von Sound auf Grundlage von Soundsample-Daten finden Sie unter „Verwenden von dynamisch generierten Audiodaten“ auf Seite 479.

Sound-Beispiel: Podcast-Player

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Podcast ist eine über das Internet verteilte Sounddatei, die entweder speziell angefordert oder abonniert wurde. Podcasts werden in der Regel als Serie veröffentlicht, die auch als ein Podcast-Kanal bezeichnet wird. Da Podcast-Episoden zwischen einer Minute und mehreren Stunden umfassen können, werden sie im Allgemeinen während der Wiedergabe gestreamt. Podcast-Episoden, die auch als Objekte bezeichnet werden, liegen in der Regel im MP3-Format vor. Auch Video-Podcasts sind sehr populär. Diese Beispielanwendung gibt jedoch nur Audio-Podcasts wieder, die MP3-Dateien verwenden.

Dieses Beispiel ist keine Podcast Aggregator-Anwendung mit vollständigem Funktionsumfang. Beispielsweise verwaltet sie keine Abonnements von bestimmten Podcasts oder speichert Aufzeichnungen darüber, welche Podcasts der Benutzer bereits gehört hat. Sie kann jedoch als Startpunkt für einen Podcast-Aggregator mit einem größeren Funktionsumfang dienen.

Das Beispiel „Podcast Player“ veranschaulicht die folgenden ActionScript-Programmiertechniken:

- Lesen eines externen RSS-Feed und Einlesen des XML-Inhalts
- Erstellen einer SoundFacade-Klasse zum einfachen Laden und Wiedergeben von Sounddateien
- Anzeigen des Fortschritts der Soundwiedergabe
- Unterbrechen und Fortsetzen der Soundwiedergabe

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „Podcast Player“ befinden sich im Ordner „Samples/PodcastPlayer“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
PodcastPlayer.mxml oder PodcastPlayer.fla	Die Benutzeroberfläche der Anwendung im Flex-Format (MXML) oder Flash-Format (FLA).
comp/example/programmingas3/podcastplayer/PodcastPlayer.as	Document-Klasse mit der Benutzeroberflächenlogik für den Podcast-Player (nur Flash).
SoundPlayer.mxml	Eine MXML-Komponente, die Wiedergabeschaltflächen und Fortschrittleisten anzeigt sowie die Soundwiedergabe steuert (nur für Flex).
main.css	Stile für die Benutzeroberfläche der Anwendung (nur Flex).
images/	Symbole zum Gestalten der Schaltflächen (nur Flex).
comp/example/programmingas3/podcastplayer/SoundPlayer.as	Klasse für das SoundPlayer-Movieclipsymbol mit der Benutzeroberflächenlogik für den Soundplayer (nur Flash).
comp/example/programmingas3/podcastplayer/PlayButtonRenderer.as	Benutzerdefinierter Zellen-Renderer für die Anzeige einer Wiedergabe-Schaltfläche in einer Zelle eines Datengitters (nur Flash).
com/example/programmingas3/podcastplayer/RSSBase.as	Eine Basisklasse, die allgemeine Eigenschaften und Methoden für die RSSChannel- und die RSSItem-Klasse bereitstellt.

Datei	Beschreibung
com/example/programmingas3/podcastplayer/RSSChannel.as	Eine ActionScript-Klasse, die Daten über einen RSS-Kanal enthält.
com/example/programmingas3/podcastplayer/RSSItem.as	Eine ActionScript-Klasse, die Daten über ein RSS-Objekt enthält.
com/example/programmingas3/podcastplayer/SoundFacade.as	Die ActionScript-Hauptklasse der Anwendung. Diese Klasse kapselt die Methoden und Ereignisse der Sound- und der SoundChannel-Klasse ein und fügt eine Unterstützung für das Unterbrechen und Fortsetzen der Wiedergabe hinzu.
com/example/programmingas3/podcastplayer/URLService.as	Eine ActionScript-Klasse, die Daten von einer Remote-URL empfängt.
playerconfig.xml	Eine XML-Datei, die eine Liste der RSS-Feeds enthält, die Podcast-Kanäle darstellen.
comp/example/programmingas3/utils/DateUtil.as	Klasse, die zur einfachen Datumsformatierung verwendet wird (nur Flash).

Lesen von RSS-Daten eines Podcast-Kanals

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zunächst liest die Anwendung „Podcast Player“ Informationen über die Anzahl an Podcast-Kanälen und deren Episoden:

1. Als Erstes liest die Anwendung eine XML-Konfigurationsdatei mit einer Liste der Podcast-Kanäle ein und zeigt diese Kanalliste dem Benutzer an.

2. Nachdem der Benutzer einen der Podcast-Kanäle ausgewählt hat, liest das Programm den RSS-Feed des Kanals ein und zeigt eine Liste der Kanalepisoden an.

In diesem Beispiel wird die URLLoader utility-Klasse dazu verwendet, um textbasierte Daten von einem remoten Speicherort oder aus einer lokalen Datei zu empfangen. Zunächst erstellt der Podcast Player ein URLLoader-Objekt, um eine Liste der RSS-Feeds im XML-Format aus der Datei „playerconfig.xml“ abzurufen. Dann, wenn der Benutzer einen bestimmten Feed in der Liste markiert, wird ein neues URLLoader-Objekte erstellt, um die RSS-Daten von der URL dieses Feeds zu lesen.

Vereinfachen des Ladens und der Soundwiedergabe mithilfe der SoundFacade-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die ActionScript 3.0-Soundarchitektur ist leistungsfähig, aber komplex. Anwendungen, die nur einfache Funktionen zum Laden und zur Wiedergabe von Sounds umfassen müssen, können mithilfe einer Klasse erstellt werden, die einige der komplexen Funktionen ausblendet, und nur einen einfachen Satz von Methoden aufruft und Ereignissen bereitstellt. In der Welt des Softwaredesigns wird eine solche Klasse als *facade* (Fassade) bezeichnet.

Die SoundFacade-Klasse stellt eine einfache Schnittstelle für die folgenden Aufgaben bereit:

- Laden von Sounddateien mithilfe eines Sound-Objekts, eines SoundLoaderContext-Objekts und der SoundMixer-Klasse

- Wiedergabe von Sounddateien mithilfe eines Sound-Objekts und eines SoundChannel-Objekts
- Auslösen von Ereignissen zum Wiedergabefortschritt
- Unterbrechen und Fortsetzen der Soundwiedergabe mithilfe eines Sound-Objekts und eines SoundChannel-Objekts

Die SoundFacade-Klasse versucht, den wichtigsten Teil der Funktionsmerkmale der ActionScript-Soundklassen bei geringerer Komplexität bereitzustellen.

Im folgenden Code wird die Deklaration der Klasse, der Klasseneigenschaften und der `SoundFacade()`-Konstruktormethode gezeigt:

```
public class SoundFacade extends EventDispatcher
{
    public var s:Sound;
    public var sc:SoundChannel;
    public var url:String;
    public var bufferTime:int = 1000;

    public var isLoaded:Boolean = false;
    public var isReadyToPlay:Boolean = false;
    public var isPlaying:Boolean = false;
    public var isStreaming:Boolean = true;
    public var autoLoad:Boolean = true;
    public var autoPlay:Boolean = true;

    public var pausePosition:int = 0;

    public static const PLAY_PROGRESS:String = "playProgress";
    public var progressInterval:int = 1000;
    public var playTimer:Timer;

    public function SoundFacade(soundUrl:String, autoLoad:Boolean = true,
                                autoPlay:Boolean = true, streaming:Boolean = true,
                                bufferTime:int = -1):void
    {
        this.url = soundUrl;

        // Sets Boolean values that determine the behavior of this object
        this.autoLoad = autoLoad;
        this.autoPlay = autoPlay;
        this.isStreaming = streaming;

        // Defaults to the global bufferTime value
        if (bufferTime < 0)
        {
            bufferTime = SoundMixer.bufferTime;
        }

        // Keeps buffer time reasonable, between 0 and 30 seconds
        this.bufferTime = Math.min(Math.max(0, bufferTime), 30000);

        if (autoLoad)
        {
            load();
        }
    }
```

Die SoundFacade-Klasse erweitert die EventDispatcher-Klasse, sodass sie eigene Ereignisse auslösen kann. Zunächst deklariert der Klassencode Eigenschaften für ein Sound- und ein SoundChannel-Objekt. Außerdem speichert die Klasse den URL-Wert der Sounddatei und eine `bufferTime`-Eigenschaft, die beim Streamen des Sounds verwendet werden. Weiterhin akzeptiert sie einige boolesche Parameterwerte, die sich auf das Verhalten beim Laden und bei der Wiedergabe auswirken:

- Der `autoLoad`-Parameter weist das Objekt an, dass der Sound geladen werden soll, sobald dieses Objekt erstellt wurde.
- Der `autoPlay`-Parameter gibt an, dass die Soundwiedergabe beginnen soll, sobald ausreichend Daten geladen wurden. Handelt es sich um einen Streaming-Sound, beginnt die Wiedergabe, sobald ausreichend Daten geladen wurden. Die erforderliche Datenmenge wird mit der Eigenschaft `bufferTime` vorgegeben.
- Der `streaming`-Parameter gibt an, dass die Wiedergabe dieser Sounddatei beginnen kann, wenn das Laden der Daten vollständig abgeschlossen wurde.

Der `bufferTime`-Parameter hat einen Standardwert von -1. Wenn die Konstruktormethode einen negativen Wert im `bufferTime`-Parameter erfasst, stellt sie die `bufferTime`-Eigenschaft auf den Wert `SoundMixer.bufferTime` ein. Damit kann die Anwendung den globalen `SoundMixer.bufferTime`-Wert als Standardwert annehmen.

Wenn der `autoLoad`-Parameter auf `true` gesetzt ist, ruft die Konstruktormethode unmittelbar die folgende `load()`-Methode auf, um das Laden der Sounddatei zu starten:

```
public function load():void
{
    if (this.isPlaying)
    {
        this.stop();
        this.s.close();
    }
    this.isLoaded = false;

    this.s = new Sound();

    this.s.addEventListener(ProgressEvent.PROGRESS, onLoadProgress);
    this.s.addEventListener(Event.OPEN, onLoadOpen);
    this.s.addEventListener(Event.COMPLETE, onLoadComplete);
    this.s.addEventListener(Event.ID3, onID3);
    this.s.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
    this.s.addEventListener(SecurityErrorEvent.SECURITY_ERROR, onIOError);

    var req:URLRequest = new URLRequest(this.url);

    var context:SoundLoaderContext = new SoundLoaderContext(this.bufferTime, true);
    this.s.load(req, context);
}
```

Die `load()`-Methode erstellt ein neues Sound-Objekt und fügt Listener für alle wichtigen Soundereignisse hinzu. Dann weist sie das Sound-Objekt an, die Sounddatei zu laden. Dabei verwendet sie ein SoundLoaderContext-Objekt, um den `bufferTime`-Wert zu übergeben.

Da die `url`-Eigenschaft geändert werden kann, können mit einer SoundFacade-Instanz verschiedene Sounddateien nacheinander wiedergegeben werden: Ändern Sie einfach die `url`-Eigenschaft und rufen Sie die `load()`-Methode auf – die neue Sounddatei wird geladen.

Die folgenden drei Ereignis-Listener-Methoden zeigen, wie das SoundFacade-Objekt den Ladefortschritt verfolgt und dann entscheidet, wann der Sound wiedergegeben wird:

```
public function onLoadOpen(event:Event):void
{
    if (this.isStreaming)
    {
        this.isReadyToPlay = true;
        if (autoPlay)
        {
            this.play();
        }
    }
    this.dispatchEvent(event.clone());
}

public function onLoadProgress(event:ProgressEvent):void
{
    this.dispatchEvent(event.clone());
}

public function onLoadComplete(event:Event):void
{
    this.isReadyToPlay = true;
    this.isLoaded = true;
    this.dispatchEvent(evt.clone());

    if (autoPlay && !isPlaying)
    {
        play();
    }
}
```

Die `onLoadOpen()`-Methode wird ausgeführt, wenn das Laden des Sounds beginnt. Wenn der Sound im Streaming-Modus wiedergegeben werden kann, stellt die `onLoadComplete()`-Methode das `isReadyToPlay`-Flag direkt auf `true` ein. Das `isReadyToPlay`-Flag legt fest, ob die Anwendung die Soundwiedergabe starten kann, eventuell als Reaktion auf eine Benutzeraktion wie das Klicken auf die Wiedergabe-Schaltfläche. Die SoundChannel-Klasse verwaltet die Pufferung der Sounddaten, es ist also nicht erforderlich, explizit zu überprüfen, ob ausreichend Daten geladen wurden, bevor die `play()`-Methode aufgerufen wird.

Die `onLoadProgress()`-Methode wird während des Ladeprozesses in regelmäßigen Abständen ausgeführt. Sie sendet einfach einen Klon ihres ProgressEvent-Objekts für den Code, der dieses SoundFacade-Objekt verwendet.

Nachdem die Sounddaten vollständig geladen wurden, wird die `onLoadComplete()`-Methode ausgeführt, die bei Bedarf die `play()`-Methode für nicht gestreamte Sounds aufruft. Die `play()`-Methode wird im Folgenden gezeigt.

```
public function play(pos:int = 0):void
{
    if (!this.isPlaying)
    {
        if (this.isReadyToPlay)
        {
            this.sc = this.s.play(pos);
            this.sc.addEventListener(Event.SOUND_COMPLETE, onPlayComplete);
            this.isPlaying = true;

            this.playTimer = new Timer(this.progressInterval);
            this.playTimer.addEventListener(TimerEvent.TIMER, onPlayTimer);
            this.playTimer.start();
        }
    }
}
```

Die `play()`-Methode ruft die `Sound.play()`-Methode auf, wenn der Sound bereit zur Wiedergabe ist. Das resultierende SoundChannel-Objekt wird in der `sc`-Eigenschaft gespeichert. Dann erstellt die `play()`-Methode ein Timer-Objekt, das in regelmäßigen Abständen Wiedergabefortschrittsereignisse auslöst.

Anzeigen des Wiedergabefortschritts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Erstellen eines Timer-Objekts zur Steuerung der Wiedergabeüberwachung ist ein komplexer Vorgang, den Sie jedoch nur einmal ausführen müssen. Durch die Kapselung dieser Timer-Logik in eine wiederverwendbare Klasse wie die SoundFacade-Klasse können Anwendungen auf die gleichen Fortschrittsereignisse überwachen, wenn ein Sound geladen und wenn er wiedergegeben wird.

Das Timer-Objekt wird von der `SoundFacade.play()`-Methode erstellt, die jede Sekunde eine TimerEvent-Instanz auslöst. Die folgende `onPlayTimer()`-Methode wird immer dann ausgeführt, wenn ein neues TimerEvent-Ereignis eintrifft:

```
public function onPlayTimer(event:TimerEvent):void
{
    var estimatedLength:int =
        Math.ceil(this.s.length / (this.s.bytesLoaded / this.s.bytesTotal));
    var progEvent:ProgressEvent =
        new ProgressEvent(PLAY_PROGRESS, false, false, this.sc.position, estimatedLength);
    this.dispatchEvent(progEvent);
}
```

Die `onPlayTimer()`-Methode implementiert die Technik zur Einschätzung der Größe, die im Abschnitt „Überwachen der Wiedergabe“ auf Seite 483 beschrieben wurde. Dann erstellt sie eine neue ProgressEvent-Instanz mit dem Ereignistyp `SoundFacade.PLAY_PROGRESS`; die `bytesLoaded`-Eigenschaft ist auf die aktuelle Position des SoundChannel-Objekts eingestellt und die `bytesTotal`-Eigenschaft auf die geschätzte Länge der Sounddaten.

Unterbrechen und Fortsetzen der Wiedergabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die im vorangegangenen Abschnitt vorgestellte `SoundFacade.play()`-Methode akzeptiert einen `pos`-Parameter, der einer Startposition in den Sounddaten entspricht. Wenn der `pos`-Wert Null beträgt, startet der Sound am Anfang der Datei.

Darüber hinaus akzeptiert die `SoundFacade.stop()`-Methode einen `pos`-Parameter. Dies wird im Folgenden gezeigt:

```
public function stop(pos:int = 0):void
{
    if (this.isPlaying)
    {
        this.pausePosition = pos;
        this.sc.stop();
        this.playTimer.stop();
        this.isPlaying = false;
    }
}
```

Immer, wenn die `SoundFacade.stop()`-Methode aufgerufen wird, stellt sie die `pausePosition`-Eigenschaft so ein, dass die Anwendung weiß, wo sie den Abspielkopf positionieren soll, wenn der Benutzer die Wiedergabe des gleichen Sounds fortsetzen möchte.

Die im Folgenden gezeigten Methoden `SoundFacade.pause()` und `SoundFacade.resume()` rufen die Methoden `SoundFacade.stop()` und `SoundFacade.play()` auf und übergeben jedes Mal einen `pos`-Parameterwert.

```
public function pause():void
{
    stop(this.sc.position);
}

public function resume():void
{
    play(this.pausePosition);
}
```

Die `pause()`-Methode übergibt den aktuellen `SoundChannel.position`-Wert an die `play()`-Methode, die diesen Wert in der `pausePosition`-Eigenschaft speichert. Die `resume()`-Methode startet die Wiedergabe des gleichen Sounds mit dem `pausePosition`-Wert als Startpunkt neu.

Aufwerten der Beispielanwendung „Podcast Player“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Dieses Beispiel stellt einen Podcast-Player mit Grundfunktionen dar und illustriert die Verwendung der wiederverwendbaren SoundFacade-Klasse. Sie können weitere Funktionen hinzufügen, um diese Anwendung aufzuwerten. Dazu gehören unter anderem folgende Funktionen:

- Speichern einer Liste der Feeds und Nutzungsinformationen über jede Episode in einer SharedObject-Instanz, die der Benutzer beim nächsten Ausführen der Anwendung aufrufen kann
- Ermöglichen, dass Benutzer eigene RSS-Feeds zur Liste der Podcast-Kanäle hinzufügen können
- Speichern der Position des Abspielkopfs, wenn der Benutzer eine Episode stoppt oder verlässt, sodass der Benutzer sie beim nächsten Ausführen der Anwendung an der gleichen Stelle fortsetzen kann
- Herunterladen von MP3-Dateien zur Offlinewiedergabe, wenn der Benutzer keine Verbindung mit dem Internet hergestellt hat
- Hinzufügen von Abonnementfunktionen, die regelmäßig auf neue Episoden in einem Podcast-Kanal prüfen und die Episodenliste automatisch aktualisieren
- Hinzufügen von Funktionen zum Suchen und Browsen nach Podcasts von einem Podcast-Hosting-Dienst wie z. B. Odeo.com

Kapitel 25: Verwenden von Videos

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Video ist eine der herausragendsten Technologien im Internet. Die traditionelle Präsentation von Video – in einem rechteckigen Fenster mit einer Fortschrittleiste und Steuerelementen darunter – ist jedoch nur eine Möglichkeit, Video einzusetzen. Mit ActionScript haben Sie einen viel genaueren Zugriff auf die Vorgänge beim Laden, Präsentieren und Wiedergeben von Video und können diese besser steuern.

Grundlagen zu Video

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine wichtige Eigenschaft von Adobe® Flash® Player und Adobe® AIR™ ist die Fähigkeit, Videoinformationen mit ActionScript so anzeigen und bearbeiten zu können, wie dies auch mit anderen visuellen Inhalten wie Bildern, Animationen, Text usw. möglich ist. Wenn Sie eine Flash Video-Datei (FLV) in Adobe Flash CS4 Professional erstellen, können Sie eine Skin auswählen, die allgemeine Wiedergabe-Steuerelemente enthält. Es besteht jedoch kein Grund, dass Sie sich auf die verfügbaren Optionen beschränken. Mit ActionScript haben Sie eine genauere Kontrolle über das Laden, Anzeigen und Abspielen von Video – mit anderen Worten, Sie können Ihre eigene Videoplayer-Skin erstellen oder Ihr Video in einer weniger traditionellen Weise einsetzen. Beim Arbeiten mit Video in ActionScript verwenden Sie eine Kombination aus verschiedenen Klassen:

- Video-Klasse: Die klassische Box mit Videomaterial auf der Bühne ist eine Instanz der Video-Klasse. Die Video-Klasse ist ein Anzeigeobjekt; sie kann also mit den gleichen Techniken bearbeitet werden, die auch für andere Anzeigeobjekte gelten: Positionieren, Anwenden von Transformationen, Anwenden von Filtern und Mischmodi usw.
- StageVideo-Klasse: Die Video-Klasse verwendet zum Dekodieren und Rendern normalerweise die Software. Wenn das Gerät die GPU-Hardwarebeschleunigung unterstützt, kann Ihre Anwendung die hardwarebeschleunigte Darstellung optimal nutzen, indem Sie zur StageVideo-Klasse wechseln. Die StageVideo-API umfasst mehrere Ereignisse, die dem Code signalisieren, wann zwischen StageVideo- und Video-Objekten hin und her gewechselt werden sollte. Beim Bühnenvideo gelten einige geringfügige Einschränkungen für das Abspielen von Video. Wenn diese Einschränkungen in Ihrer Anwendung hinnehmbar sind, sollten Sie die StageVideo-API implementieren. Einzelheiten finden Sie unter „Richtlinien und Einschränkungen“ auf Seite 547.
- NetStream-Klasse: Wenn Sie eine Videodatei zur Steuerung durch ActionScript laden, repräsentiert eine NetStream-Instanz die Quelle der Videoinhalte – in diesem Fall ein Videodatenstream. Das Verwenden einer NetStream-Instanz erfordert ein NetConnection-Objekt, das die Verbindung zur Videodatei darstellt – wie ein Tunnel, durch den die Videodaten zugeführt werden.
- Camera-Klasse: Wenn Sie mit Videodaten von einer Kamera arbeiten, die an den Benutzercomputer angeschlossen ist, stellt eine Camera-Instanz die Quelle der Videoinhalte dar – die Benutzerkamera und die von ihr zur Verfügung gestellten Videodaten. Als Neuerung in Flash Player 11.4 und AIR 3.4 können Sie eine Kamera verwenden, um StageVideo zuzuführen.

Wenn Sie externe Videos laden, können Sie die Datei von einem Standard-Webserver laden, um den Inhalt progressiv wiederzugeben, oder Sie verwenden Streaming-Video, das von einem speziellen Server wie z. B. Flash® Media Server von Adobe bereitgestellt wird.

Wichtige Konzepte und Begriffe

Cue-Point Eine Markierung, die an einem bestimmten Zeitpunkt in eine Videodatei eingefügt werden kann, beispielsweise als Lesezeichen für einen bestimmten Punkt im Video oder um zusätzliche Daten bereitzustellen, die sich auf diesen Augenblick im Video beziehen.

Kodierung Die Umwandlung von Videodaten aus einem Format in ein anderes Videodatenformat, beispielsweise ein Quellvideo in einer hohen Auflösung, das zur Verwendung im Internet in ein Format mit einer weniger hohen Auflösung umgewandelt wird.

Bild Ein einzelnes Segment mit Videoinformationen. Jedes Bild (häufig wird auch der englische Begriff „Frame“ verwendet) ist im Grunde ein Standbild, das eine Momentaufnahme eines bestimmten Augenblicks darstellt. Durch die schnelle Wiedergabe der Bilder nacheinander entsteht der Eindruck einer Bewegung.

Schlüsselbild Ein Videobild, in dem die vollständigen Informationen zum Bild enthalten sind. Andere Bilder, die dem Schlüsselbild (häufig wird auch der englische Begriff Keyframe verwendet) folgen, enthalten nur Informationen darüber, worin sie sich vom Schlüsselbild unterscheiden, und nicht mehr die gesamten Bildinformationen.

Metadaten Informationen zu einer Videodatei, die in die Videodatei eingebettet sind und wieder abgerufen werden, wenn die Videodatei geladen ist.

Progressiver Download Wenn eine Videodatei von einem Standardwebserver bereitgestellt wird, werden die Videodaten progressiv, das heißt sequenziell, heruntergeladen. Dies hat den Vorteil, dass die Videowiedergabe bereits beginnen kann, noch bevor die gesamte Datei heruntergeladen wurde. Andererseits können Sie nicht zu einem Teil im Video springen, der noch nicht geladen wurde.

Streaming Eine Alternative zum progressiven Download. Hierbei stellt ein spezieller Videoserver Videodaten über das Internet bereit, wobei eine Technologie namens Streaming (manchmal auch als „true Streaming“ bezeichnet) genutzt wird. Beim Streaming lädt der anzeigende Computer niemals die gesamte Videodatei herunter. Um die Downloadzeiten in einem akzeptablen Rahmen zu halten, lädt der Computer immer nur einen Teil der gesamten Videoinformationen herunter. Da ein spezieller Server die Bereitstellung der Videoinhalte überwacht, kann auf jeden Teil des Videos zugegriffen werden. Es ist nicht erforderlich, dass die gesamten Daten heruntergeladen werden, bevor darauf zugegriffen wird.

Videoformate

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zusätzlich zum Adobe-FLV-Videoformat unterstützen Flash Player und Adobe AIR Video und Audio, das in H.264 und HE-AAC aus den MPEG-4-Standarddateiformaten kodiert ist. Diese Formate streamen Video in hoher Qualität bei niedrigen Bitraten. Entwickler können branchenübliche Standard-Tools, zum Beispiel Adobe Premiere Pro und Adobe After Effects, für die Erstellung und Bereitstellung von Videoinhalten nutzen.

Typ	Format	Container
Video	H.264	MPEG-4: MP4, M4V, F4V, 3GPP
Video	Sorenson Spark	FLV-Datei
Video	ON2 VP6	FLV-Datei
Audio	AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4: MP4, M4V, F4V, 3GPP

Typ	Format	Container
Audio	MP3	MP3
Audio	Nellymoser	FLV-Datei
Audio	Speex	FLV-Datei

Verwandte Hilfethemen

Flash Media Server: Unterstützte Codecs

Adobe HTTP Dynamic Streaming

Kodieren von Video für Mobilgeräte

AIR unter Android kann eine breite Palette an H.264-Videos dekodieren. Nur ein kleiner Teil von H.264-Videos kann jedoch auf Mobiltelefonen unterbrechungsfrei wiedergegeben werden, da zahlreiche Mobiltelefone nur über eingeschränkte Verarbeitungsleistung verfügen. Adobe Flash Player für Mobilgeräte kann H.264-Videos mit der integrierten Hardwarebeschleunigung dekodieren. Diese Dekodierung liefert eine höhere Qualität bei geringerem Leistungsbedarf.

Der H.264-Standard unterstützt mehrere Kodierungstechniken. Nur hochwertige Geräte können Videos mit komplexen Profilen und Leveln problemlos abspielen. Doch die meisten Geräte können Video abspielen, das im Baseline-Profil kodiert ist. Die Hardwarebeschleunigung ist auf Mobilgeräten für einen Teil dieser Techniken verfügbar. Die profile- und level-Parameter definieren diese Teilmenge der Kodierungstechniken und Einstellungen, die vom Encoder verwendet werden. Entwickler müssen das Video in einer ausgewählten Auflösung kodieren, die für die meisten Geräte gut geeignet ist.

Welche Auflösungen von der Hardwarebeschleunigung profitieren, ist zwar von Gerät zu Gerät verschieden, doch die meisten Geräte unterstützen die folgenden Standardauflösungen.

Seitenverhältnis	Empfohlene Auflösungen		
4:3	640 × 480	512 × 384	480 × 360
16:9	640 × 360	512 x 288	480 × 272

Hinweis: *Flash Player unterstützt alle Level und Profile des H.264-Standards. Diese Empfehlungen führen auf den meisten Geräten zur optimalen Nutzung der Hardwarebeschleunigung und zu einer höheren Benutzerfreundlichkeit. Diese Empfehlungen müssen aber nicht zwingend beachtet werden.*

Ausführliche Erläuterungen und Informationen zu den Kodierungseinstellungen in Adobe Media Encoder CS5 finden Sie unter Recommendations for encoding H.264 video for Flash Player 10.1 on mobile devices.

Hinweis: *Unter iOS kann nur Video, das mit den Sorenson Spark- und On2 VP6-Codecs kodiert wurde, mit der Video-Klasse abgespielt werden. Sie können H.264-kodiertes Video im Videoplayer des Geräts abspielen, indem Sie die URL zum Video über die* `flash.net.navigateToURL()`*-Funktion aufrufen. Außerdem können Sie H.264-Video über das <video>-Tag in einer HTML-Seite abspielen, die in einem StageWebView-Objekt angezeigt wird.*

Kompatibilität von Flash Player und AIR mit kodierten Videodateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player 7 unterstützt mit dem Sorenson™ Spark™-Videocodec kodierte FLV-Dateien. Flash Player 8 unterstützt FLV-Dateien, die mit dem Sorenson Spark- oder On2 VP6-Encoder in Flash Professional 8 kodiert wurden. Der On2 VP6-Video-Codec unterstützt einen Alphakanal.

Flash Player 9.0.115.0 und neuere Versionen unterstützen Dateien, die aus dem MPEG-4-Standardcontainerformat abgeleitet werden. Zu diesen Dateien gehören F4V, MP4, M4A, MOV, MP4V, 3GP und 3G2, wenn sie H.264-Video oder mit HE-AAC v2 kodiertes Audio oder beides enthalten. Mit H.264 ist die Videoqualität im Vergleich zu demselben Codierungsprofil in Sorenson oder On2 bei niedrigeren Bitraten höher. HE-AAC v2 ist eine Erweiterung des AAC-Formats (ein Standardaudioformat, das im MPEG-4-Videostandard definiert ist). HE-AAC v2 verwendet Spectral Band Replication (SBR) und Parametric Stereo (PS), damit bei niedrigeren Bitraten effizienter kodiert werden kann.

In der folgenden Tabelle sind die unterstützten Codecs aufgeführt. Außerdem werden die entsprechenden SWF-Dateiformate und die für die Wiedergabe erforderlichen Versionen von Flash Player und AIR genannt:

Codec	SWF-Dateiformatversion (älteste unterstützte Version)	Flash Player und AIR (älteste für die Wiedergabe erforderliche Version)
Sorenson Spark	6	Flash Player 6, Flash Lite 3
On2 VP6	6	Flash Player 8, Flash Lite 3. Nur Flash Player 8 und neuere Versionen unterstützen die Veröffentlichung und die Wiedergabe von On6 VP-Video.
H.264 (MPEG-4 Part 10)	9	Flash Player 9 Update 3, AIR 1.0
ADPCM	6	Flash Player 6, Flash Lite 3
MP3	6	Flash Player 6, Flash Lite 3
AAC (MPEG-4 Part 3)	9	Flash Player 9 Update 3, AIR 1.0
Speex (Audio)	10	Flash Player 10, AIR 1.5
Nellymoser	6	Flash Player 6

Die Adobe F4V- und FLV-Videodateiformate

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Adobe bietet die F4V- und FLV-Videodateiformate für das Streaming von Inhalten an Flash Player und AIR. Eine umfassende Beschreibung dieser Videodateiformate finden Sie unter www.adobe.com/go/video_file_format_de.

Das F4V-Videodateiformat

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ab Flash Player Update 3 (9.0.115.0) und AIR 1.0 unterstützen Flash Player und AIR das Adobe F4V-Videoformat, das auf dem ISO MP4-Format basiert. Teilsätze des Formats unterstützen verschiedene Funktionen. Flash Player erwartet als Anfang einer gültigen F4V-Datei eines der folgenden Atome der obersten Ebene:

- ftyp

  Das ftyp-Atom gibt die Funktionen an, die ein Programm unterstützen muss, damit ein bestimmtes Dateiformat wiedergegeben werden kann.

- moov

  Das moov-Atom ist praktisch der Header einer F4V-Datei. Es enthält eines oder mehrere andere Atome, die wiederum andere Atome enthalten, mit denen die Struktur der F4V-Daten definiert wird. Eine F4V-Datei muss ein moov-Atom aber nicht mehr enthalten.

- mdat

  Ein mdat-Atom enthält die Daten der F4V-Datei. Ein F4V-Datei enthält nur ein mdat-Atom. Die Datei muss auch ein moov-Atom enthalten, da das mdat-Atom alleine nicht verstanden werden kann.

F4V-Dateien unterstützen Multibyte-Ganzzahlen in big-endian-Byte-Reihenfolge, bei der das höchstwertige Byte an erster Stelle bzw. der niedrigsten Adresse steht.

Das FLV-Videodateiformat

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Adobe FLV-Dateiformat enthält kodierte Audio- und Videodaten, die mit Flash Player wiedergegeben werden können. Mit einem Encoder wie Adobe Media Encoder oder Sorenson™ Squeeze können Sie ein QuickTime- oder Windows Media-Video in eine FLV-Datei konvertieren.

Hinweis: *Sie erstellen eine FLV-Datei, indem Sie Video in Flash importieren und als eine FLV-Datei exportieren. Flash verfügt über ein Plug-In für den FLV-Export, mit dem Sie FLV-Dateien aus unterstützten Videobearbeitungsprogrammen exportieren können. Um FLV-Dateien von einem Webserver zu laden, müssen Sie die Dateinamenerweiterung und den MIME-Typ beim Webserver registrieren. Lesen Sie dazu Ihre Webserverdokumentation. Der MIME-Typ für FLV-Dateien lautet* `video/x-flv`*. Weitere Informationen finden Sie unter „Konfigurieren von FLV-Dateien für das Hosten auf einem Server“ auf Seite 538.*

Weitere Informationen zu FLV-Dateien finden Sie unter „Erweiterte Themen für Videodateien“ auf Seite 537.

Externe im Vergleich zu eingebetteten Videos

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei externen Videodateien ergeben sich gewisse Möglichkeiten, die bei importierten Videodateien nicht zur Verfügung stehen:

- In Ihrer Anwendung können Sie längere Videoclips einsetzen, ohne dass sich dies negativ auf die Abspielgeschwindigkeit auswirkt. Externe Videodateien verwenden Cache-Speicher, was bedeutet, dass große Dateien in kleinen Teilen gespeichert werden und dass dynamisch auf sie zugegriffen wird. Aus diesem Grund benötigen externe F4V- und FLV-Dateien weniger Speicher als eingebettete Videodateien.
- Die Bildraten externer Videodateien können sich von denen der SWF-Dateien, die sie wiedergeben, unterscheiden. Sie können z. B. die Bildrate der SWF-Datei auf 30 Bps (Bilder pro Sekunde) und die des Videos auf 21 Bps einstellen. Mit dieser Einstellung haben Sie eine bessere Kontrolle über das Video als bei einem eingebetteten Video und können so eine flüssige Videowiedergabe erzielen. Sie können somit Videodateien auch mit anderen Bildraten anzeigen, ohne den vorhandenen SWF-Dateiinhalt ändern zu müssen.
- Bei externen Videodateien wird die Wiedergabe des SWF-Inhalts nicht unterbrochen, während die Videodatei geladen wird. Importierte Videodateien unterbrechen gelegentlich die Dokumentwiedergabe, um bestimmte Funktionen auszuführen, z. B., um auf ein CD-ROM-Laufwerk zuzugreifen. Videodateien können Funktionen unabhängig vom SWF-Inhalt ausführen, ohne die Wiedergabe zu unterbrechen.
- Mit externen FLV-Dateien können Videoinhalte einfacher mit Untertiteln versehen werden, da Sie mit Ereignisprozeduren auf die Metadaten des Videos zugreifen können.

Video-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Video-Klasse können Sie Live-Streaming-Videos in einer Anwendung wiedergeben, ohne sie in die SWF-Datei einbetten zu müssen. Sie können Live-Videos über die `Camera.getCamera()`-Methode aufnehmen und wiedergeben. Darüber hinaus können Sie mithilfe der Video-Klasse Videodateien (FLV-Dateien) über HTTP oder das lokale Dateisystem wiedergeben. Video kann auf verschiedene Arten in Ihren Projekten eingesetzt werden:

- Dynamisches Laden einer Videodatei mithilfe der NetConnection- und NetStream-Klasse und Anzeigen von Video in einem Video-Objekt.
- Erfassen des Videoeingangs einer Benutzerkamera. Weitere Informationen hierzu finden Sie unter „Arbeiten mit Kameras“ auf Seite 554.
- Verwenden der FLVPlayback-Komponente.
- Verwenden der VideoDisplay-Steuerung.

Hinweis: *Instanzen eines Video-Objekts auf der Bühne sind Instanzen der Video-Klasse.*

Obwohl die Video-Klasse im flash.media-Paket enthalten ist, erbt sie von der flash.display.DisplayObject-Klasse. Aus diesem Grund können alle Funktionsmerkmale, die für Anzeigeobjekte gelten, z. B. Matrixtransformationen und Filter, auch auf Video-Instanzen angewendet werden.

Weitere Informationen finden Sie unter „Bearbeiten von Anzeigeobjekten“ auf Seite 185, „Verwenden von geometrischen Objekten“ auf Seite 224 und „Anwenden von Filtern auf Anzeigeobjekte“ auf Seite 286.

Laden von Videodateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der Vorgang zum Laden von Videos mithilfe der NetStream- und NetConnection-Klasse umfasst mehrere Schritte. Da sich dies bewährt hat, wird empfohlen, die Schritte zum Hinzufügen des Video-Objekts zur Anzeigeliste, zum Hinzufügen des NetStream-Objekts zur Video-Instanz und zum Aufrufen der `play()`-Methode des NetStream-Objekts in der angegebenen Reihenfolge auszuführen:

1. NetConnection-Objekt erstellen. Wenn Sie eine Verbindung zu einer lokalen Videodatei oder zu einer Datei, die keinen Server wie zum Beispiel Adobes Flash Media Server 2 verwendet, herstellen, übergeben Sie `null` an die `connect()`-Methode, um die Videodateien von einer HTTP-Adresse oder einem lokalen Laufwerk abzuspielen. Wenn Sie eine Verbindung zu einem Server herstellen, setzen Sie den Parameter auf den URI der Anwendung auf dem Server, die die Videodatei enthält.

   ```
   var nc:NetConnection = new NetConnection();
   nc.connect(null);
   ```

2. Erstellen Sie ein neues Video-Objekt, das das Video anzeigt, und fügen Sie es zur Anzeigeliste der Bühne hinzu wie im folgenden Codefragment:

   ```
   var vid:Video = new Video();
   addChild(vid);
   ```

3. Erstellen Sie ein NetStream-Objekt und übergeben Sie dabei das NetConnection-Objekt als ein Argument an den Konstruktor. Das folgende Codefragment verbindet ein NetStream-Objekt mit der NetConnection-Instanz und richtet die Ereignisprozeduren für den Stream ein:

```
var ns:NetStream = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS,netStatusHandler);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);

function netStatusHandler(event:NetStatusEvent):void
{
    // handle netStatus events, described later
}

function asyncErrorHandler(event:AsyncErrorEvent):void
{
    // ignore error
}
```

4 Fügen Sie das NetStream-Objekt zum Video-Objekt hinzu, indem Sie die `attachNetStream()`-Methode des Video-Objekts verwenden wie im folgenden Codefragment:

```
vid.attachNetStream(ns);
```

5 Rufen Sie die `play()`-Methode des NetStream-Objekts mit der URL der Videodatei als Argument auf, um die Videowiedergabe zu starten. Mit dem folgenden Codefragment wird die Videodatei „video.mp4“ in demselben Verzeichnis wie die SWF-Datei geladen und abgespielt:

```
ns.play("video.mp4");
```

Verwandte Hilfethemen

Flex: Spark VideoPlayer control

spark.components.VideoDisplay

Steuern der Videowiedergabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die NetStream-Klasse enthält vier Hauptklassen zur Steuerung der Videowiedergabe:

`pause()`: Hält die Wiedergabe eines Videostreams an. Ist die Wiedergabe bereits angehalten, hat der Aufruf dieser Methode keine Auswirkungen.

`resume()`: Setzt die Wiedergabe eines angehaltenen Videostreams fort. Läuft die Wiedergabe bereits, hat der Aufruf dieser Methode keine Auswirkungen.

`seek()`: Sucht das Schlüsselbild, das der angegebenen Position am nächsten kommt (ein Offset in Sekunden ab dem Beginn des Streams).

`togglePause()`: Hält die Wiedergabe eines Streams an oder setzt sie fort.

Hinweis: *Es gibt keine `stop()`-Methode. Um einen Stream zu stoppen, müssen Sie in die Wiedergabe anhalten und nach dem Anfang des Video-Streams suchen.*

Hinweis: *Die `play()`-Methode setzt die Wiedergabe nicht fort, sie wird stattdessen zum Laden von Videodateien verwendet.*

Im folgenden Beispiel wird veranschaulicht, wie Sie ein Video mit zwei Schaltflächen steuern können. Um das folgende Beispiel auszuführen, erstellen Sie ein neues Dokument, und fügen Sie im Arbeitsbereich vier Schaltflächeninstanzen ein (`pauseBtn`, `playBtn`, `stopBtn` und `togglePauseBtn`):

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);
ns.play("video.flv");
function asyncErrorHandler(event:AsyncErrorEvent):void
{
    // ignore error
}

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

pauseBtn.addEventListener(MouseEvent.CLICK, pauseHandler);
playBtn.addEventListener(MouseEvent.CLICK, playHandler);
stopBtn.addEventListener(MouseEvent.CLICK, stopHandler);
togglePauseBtn.addEventListener(MouseEvent.CLICK, togglePauseHandler);

function pauseHandler(event:MouseEvent):void
{
    ns.pause();
}
function playHandler(event:MouseEvent):void
{
    ns.resume();
}
function stopHandler(event:MouseEvent):void
{
    // Pause the stream and move the playhead back to
    // the beginning of the stream.
    ns.pause();
    ns.seek(0);
}
function togglePauseHandler(event:MouseEvent):void
{
    ns.togglePause();
}
```

Durch Klicken auf die Schaltflächeninstanz `pauseBtn` während der Videowiedergabe wird die Videodatei angehalten. Ist die Wiedergabe bereits angehalten, hat das Klicken auf diese Schaltfläche keine Auswirkungen. Durch Klicken auf die Schaltflächeninstanz `playBtn` wird die Videowiedergabe fortgesetzt, wenn sie zuvor angehalten wurde, andernfalls (wenn das Video bereits wiedergegeben wird) hat das Klicken auf diese Schaltfläche keine Auswirkungen.

Erfassen des Endes eines Video-Streams

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um den Anfang und das Ende eines Video-Streams zu überwachen, müssen Sie der NetStream-Instanz einen Ereignis-Listener hinzufügen, der das Auftreten eines `netStatus`-Ereignisses überwacht. Im folgenden Code wird veranschaulicht, wie während der Videowiedergabe das Auftreten verschiedener Codes überwacht wird:

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void
{
    trace(event.info.code)
}
```

Mit dem vorangegangenen Code wird die folgende Ausgabe erzeugt:

```
NetStream.Play.Start
NetStream.Buffer.Empty
NetStream.Buffer.Full
NetStream.Buffer.Empty
NetStream.Buffer.Full
NetStream.Buffer.Empty
NetStream.Buffer.Full
NetStream.Buffer.Flush
NetStream.Play.Stop
NetStream.Buffer.Empty
NetStream.Buffer.Flush
```

Die zwei Ereignisse, deren Auftreten Sie überwachen, sind NetStream.Play.Start und NetStream.Play.Stop, die den Anfang bzw. das Ende der Videowiedergabe kennzeichnen. Im folgenden Codeausschnitt werden diese beiden Codes mit einer switch-Anweisung gefiltert und eine Nachricht gesendet:

```
function statusHandler(event:NetStatusEvent):void
{
    switch (event.info.code)
    {
        case "NetStream.Play.Start":
            trace("Start [" + ns.time.toFixed(3) + " seconds]");
            break;
        case "NetStream.Play.Stop":
            trace("Stop [" + ns.time.toFixed(3) + " seconds]");
            break;
    }
}
```

Durch Überwachen des `netStatus`-Ereignisses (`NetStatusEvent.NET_STATUS`) können Sie einen Videoplayer erstellen, der das nächste Video in einer Wiedergabeliste lädt, wenn die Wiedergabe der aktuellen Videodatei beendet ist.

Videowiedergabe im Vollbildmodus

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player und AIR ermöglichen Ihnen, eine Vollbildanwendung für Ihre Videowiedergabe zu erstellen, und unterstützen das Skalieren von Video zum Vollbild.

Bei AIR-Inhalt, der im Vollbildmodus auf dem Desktop ausgeführt wird, sind die Bildschirmschoner- und Energiesparoptionen des Systems während der Wiedergabe deaktiviert, bis der Videoeingang stoppt oder der Benutzer den Vollbildmodus beendet.

Ausführliche Informationen zur Verwendung des Vollbildmodus finden Sie unter „Verwenden des Vollbildmodus“ auf Seite 178.

Aktivieren des Vollbildmodus für Flash Player in einem Browser

Bevor Sie den Vollbildmodus für Flash Player in einem Browser implementieren können, aktivieren Sie ihn mithilfe der Veröffentlichungsvorlage für Ihre Anwendung. Vorlagen, die den Vollbildmodus zulassen, verfügen über `<object>`- und `<embed>`-Tags, die einen `allowFullScreen`-Parameter enthalten. Im folgenden Beispiel wird der `allowFullScreen`-Parameter in einem `<embed>`-Tag gezeigt.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    id="fullScreen" width="100%" height="100%"
    codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.cab">
    ...
    <param name="allowFullScreen" value="true" />
    <embed src="fullScreen.swf" allowFullScreen="true" quality="high" bgcolor="#869ca7"
        width="100%" height="100%" name="fullScreen" align="middle"
        play="true"
        loop="false"
        quality="high"
        allowScriptAccess="sameDomain"
        type="application/x-shockwave-flash"
        pluginspage="http://www.adobe.com/go/getflashplayer">
    </embed>
    ...
</object>
```

Wählen Sie in Flash „Datei“ > „Einstellungen für Veröffentlichungen“ und im Dialogfeld „Einstellungen für Veröffentlichungen“ auf der Registerkarte „HTML“ die Vorlage „Nur Flash - Vollbild zulassen“.

Stellen Sie in Flex sicher, dass die HTML-Vorlage `<object>`- und `<embed>`-Tags enthält, die den Vollbildmodus unterstützen.

Initiieren des Vollbildmodus

Wenn Flash Player-Inhalt in einem Browser ausgeführt wird, initiieren Sie den Vollbildmodus in Reaktion auf einen Mausklick oder einen Tastendruck. Sie können den Vollbildmodus zum Beispiel aufrufen, wenn der Benutzer auf eine mit „Vollbild“ beschriftete Schaltfläche klickt oder einen entsprechenden Befehl im Kontextmenü auswählt. Fügen Sie dem Objekt, bei dem die Aktion erfolgt, einen Ereignis-Listener hinzu, um auf die Benutzeraktion zu reagieren. Mit dem folgenden Code wird einer Schaltfläche, mit der der Benutzer in den Vollbildmodus wechseln kann, ein Ereignis-Listener hinzugefügt:

```
var fullScreenButton:Button = new Button();
fullScreenButton.label = "Full Screen";
addChild(fullScreenButton);
fullScreenButton.addEventListener(MouseEvent.CLICK, fullScreenButtonHandler);

function fullScreenButtonHandler(event:MouseEvent)
{
    stage.displayState = StageDisplayState.FULL_SCREEN;

}
```

Der Code initiiert den Vollbildmodus, indem die `Stage.displayState`-Eigenschaft auf `StageDisplayState.FULL_SCREEN` eingestellt wird. Dieser Code skaliert die gesamte Bühne zum Vollbild, wobei das Video proportional zum Platz, den es auf der Bühne einnimmt, skaliert wird.

Mit der `fullScreenSourceRect`-Eigenschaft können Sie einen bestimmten Bereich der Bühne zum Vollbild skalieren. Definieren Sie zunächst das Rechteck, das Sie zum Vollbild skalieren möchten. Weisen Sie es dann der `Stage.fullScreenSourceRect`-Eigenschaft zu. Mit dieser Version der `fullScreenButtonHandler()`-Funktion werden zwei zusätzliche Codezeilen hinzugefügt, die das Video zum Vollbild skalieren.

```
private function fullScreenButtonHandler(event:MouseEvent)
{
    var screenRectangle:Rectangle = new Rectangle(video.x, video.y, video.width, video.height);
    stage.fullScreenSourceRect = screenRectangle;
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

In diesem Beispiel wird zwar eine Ereignisprozedur als Reaktion auf einen Mausklick ausgelöst, die Technik zum Aktivieren des Vollbildmodus ist in Flash Player und AIR jedoch identisch. Definieren Sie das Rechteck, das Sie skalieren möchten, und stellen Sie dann die `Stage.displayState`-Eigenschaft ein. Weitere Informationen finden Sie im Handbuch ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Im nachstehenden vollständigen Beispiel wird Code hinzugefügt, mit dem die Verbindung erstellt wird und das NetStream-Objekt für das Video mit dem Abspielen beginnt.

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.media.Video;
    import flash.display.StageDisplayState;
    import fl.controls.Button;
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.events.FullScreenEvent;
    import flash.geom.Rectangle;

    public class FullScreenVideoExample extends Sprite
    {
        var fullScreenButton:Button = new Button();
        var video:Video = new Video();

        public function FullScreenVideoExample()
        {
            var videoConnection:NetConnection = new NetConnection();
            videoConnection.connect(null);

            var videoStream:NetStream = new NetStream(videoConnection);
            videoStream.client = this;

            addChild(video);

            video.attachNetStream(videoStream);

            videoStream.play("http://www.helpexamples.com/flash/video/water.flv");

            fullScreenButton.x = 100;
```

```
            fullScreenButton.y = 270;
            fullScreenButton.label = "Full Screen";
            addChild(fullScreenButton);
            fullScreenButton.addEventListener(MouseEvent.CLICK, fullScreenButtonHandler);
        }

        private function fullScreenButtonHandler(event:MouseEvent)
        {
            var screenRectangle:Rectangle = new Rectangle(video.x, video.y, video.width,
video.height);
            stage.fullScreenSourceRect = screenRectangle;
            stage.displayState = StageDisplayState.FULL_SCREEN;
        }

        public function onMetaData(infoObject:Object):void
        {
            // stub for callback function
        }
    }
}
```

Die `onMetaData()`-Funktion ist eine Rückruffunktion für die Verarbeitung von Videometadaten, sofern diese vorhanden sind. Eine Rückruffunktion ist eine Funktion, die die Laufzeitumgebung als Reaktion auf ein bestimmtes Vorkommnis oder Ereignis aufruft. In diesem Beispiel ist die `onMetaData()`-Funktion ein Teil, das die Anforderungen zum Bereitstellen der Funktion erfüllt. Weitere Informationen finden Sie unter „Schreiben von Rückrufmethoden für Metadaten und Cue-Points“ auf Seite 518.

Beenden des Vollbildmodus

Ein Benutzer kann den Vollbildmodus mit bestimmten Tastaturbefehlen, zum Beispiel mit der Escape-Taste, beenden. In ActionScript können Sie den Vollbildmodus beenden, indem Sie die `Stage.diplayState`-Eigenschaft auf `StageDisplayState.NORMAL` setzen. Mit dem Code im folgenden Beispiel wird der Vollbildmodus beendet, wenn das NetStream.Play.Stop`netStatus`-Ereignis eintritt.

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);

private function netStatusHandler(event:NetStatusEvent)
{
    if(event.info.code == "NetStream.Play.Stop")
        stage.displayState = StageDisplayState.NORMAL;
}
```

Vollbild-Hardwarebeschleunigung

Wenn Sie einen rechteckigen Bereich der Bühne zum Vollbild skalieren, verwenden Flash Player und AIR die Hardwarebeschleunigung, sofern diese verfügbar und aktiviert ist. Die Laufzeitumgebung verwendet den Grafikadapter des Computers, um die Skalierung des Videos oder eines Teils der Bühne zum Vollbild zu beschleunigen. Unter diesen Bedingungen können Flash Player-Anwendungen häufig davon profitieren, wenn von der Video-Klasse (oder Camera-Klasse, nur ab Version 11.4/AIR 3.4) zur StageVideo-Klasse gewechselt wird.

Weitere Informationen zur Hardwarebeschleunigung im Vollbildmodus finden Sie unter „Verwenden des Vollbildmodus“ auf Seite 178. Weitere Informationen zu StageVideo finden Sie unter „Verwenden der StageVideo-Klasse für die hardwarebeschleunigte Darstellung“ auf Seite 545.

Streamen von Videodateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Streamen von Dateien von einem Flash Media Server können Sie die NetConnection- und die NetStream-Klasse verwenden, um eine Verbindung zu einer Instanz eines Remote-Servers herzustellen und einen bestimmten Stream wiederzugeben. Den RTMP-Server (Real-Time Messaging Protocol) geben Sie durch Übergeben der gewünschten RTMP-URL (z. B. „rtmp://localhost/appName/appInstance“) an die `NetConnection.connect()`-Methode an, die Sie anstelle von `null` übergeben. Um einen bestimmten aufgezeichneten oder Live-Stream über den angegebenen Flash Media Server wiederzugeben, übergeben Sie an die `NetStream.play()`-Methode zur Wiedergabe den Namen der aufgezeichneten Datei oder einen identifizierenden Namen für die mit `NetStream.publish()` veröffentlichten Live-Daten.

Senden von Videos an einen Server

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie komplexere Anwendungen mit Video- oder Camera-Objekten erstellen möchten, bietet Flash Media Server eine Kombination aus Streaming Media-Funktionen und einer Entwicklungsumgebung, sodass Sie Medienanwendungen für ein breites Publikum erstellen und bereitstellen können. Mit dieser Kombination können Entwickler Anwendungen wie Video on Demand, Live-Web-Event-Broadcasts und MP3-Streaming sowie Video-Blogs, Video-Messaging und Multimedia-Chat-Umgebungen erstellen. Weitere Informationen finden Sie in der Dokumentation zu Flash Media Server unter www.adobe.com/go/learn_fms_docs_de.

Cue-Points

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Während der Kodierung können Sie Cue-Points in eine Adobe F4V- oder FLV-Videodatei einbetten. In der Vergangenheit wurden Cue-Points in Filme eingebettet, um Filmvorführer durch ein optisches Signal auf das nahende Ende der Filmspule hinzuweisen. In den Adobe F4V- und FLV-Videoformaten ermöglicht ein Cue-Point bei seinem Auftreten im Video-Stream das Auslösen von einer oder mehreren Aktionen.

Sie können mit Flash Video eine Reihe unterschiedlicher Cue-Points verwenden. Über ActionScript-Code können Sie mit Cue-Points interagieren, die Sie beim Erstellen in eine Videodatei einbetten.

- Navigation-Cue-Points: Sie fügen beim Kodieren der Videodatei Cue-Points für die Navigation in den Videodatenstrom und das Metadatenpaket ein. Mithilfe von Cue-Points für die Navigation können Benutzer zu einem bestimmten Teil einer Datei springen.
- Ereignis-Cue-Points: Sie fügen Ereignis-Cue-Points beim Kodieren der Videodatei in den Videodatenstrom und das Metadatenpaket ein. Sie können Code für die Verarbeitung von Ereignissen verfassen, die an festgelegten Punkten während der Videowiedergabe ausgelöst werden.

- ActionScript-Cue-Points: ActionScript-Cue-Points sind nur für die Flash-FLVPlayback-Komponente verfügbar. ActionScript-Cue-Points sind externe Cue-Points, die Sie mit ActionScript erstellen und darauf zugreifen. Sie können Code verfassen, mit dem diese Cue-Points in Abhängigkeit von der Videowiedergabe ausgelöst werden. Diese Cue-Points sind weniger genau als die eingebetteten Cue-Points (bis zu einer Zehntelsekunde Abweichung), da sie im Video Player separat verwaltet werden. Wenn Sie eine Anwendung erstellen möchten, in der Benutzer Cue-Points anwählen können, sollten Sie die Cue-Points beim Kodieren der Datei einbetten und keine ActionScript-Cue-Points verwenden. Es empfiehlt sich, die Cue-Points in die FLV-Datei einzubetten, da sie dann exakter sind.

Bei Cue-Points für die Navigation wird an der angegebenen Cue-Point-Position ein Schlüsselbild erstellt. Mit ActionScript-Code können Sie dann die Wiedergabeposition eines Video Players auf diese Stelle setzen. Sie können bestimmte Punkte in einer Videodatei festlegen, an die Benutzer dann springen können. Wenn das Video beispielsweise aus mehreren Kapiteln oder Segmenten besteht, können Sie die Videowiedergabe steuern, indem Sie in die Videodatei Cue-Points für die Navigation einbetten.

Weitere Informationen zum Kodieren von Adobe Videodateien mit Cue-Points finden Sie unter „Cue-Points einbetten“ im Handbuch *Flash verwenden*.

Mit ActionScript-Code können Sie auf Cue-Point-Parameter zugreifen. Cue-Point-Parameter sind Bestandteil des Ereignisobjekts, das von der Rückrufprozedur empfangen wird.

Mithilfe der Ereignisprozedur `NetStream.onCuePoint` können Sie bestimmte Aktionen im Code auslösen, wenn eine FLV-Datei einen bestimmten Cue-Point erreicht.

Um eine Aktion mit einem Cue-Point in einer F4V-Videodatei zu synchronisieren, müssen Sie die Cue-Point-Daten entweder mit der `onMetaData()`- oder der `onXMPData()`-Rückruffunktion abrufen. Anschließend müssen Sie den Cue-Point mit der Timer-Klasse in ActionScript 3.0 auslösen. Weitere Informationen zu F4V-Cue-Points finden Sie unter „Verwenden von onXMPData()“ auf Seite 530.

Weitere Informationen zum Umgang mit Cue-Points und Metadaten finden Sie unter „Schreiben von Rückrufmethoden für Metadaten und Cue-Points“ auf Seite 518.

Schreiben von Rückrufmethoden für Metadaten und Cue-Points

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn bestimmte Cue-Points erreicht oder bestimmte Metadaten vom Player empfangen werden, können Sie Aktionen in Ihrer Anwendung auslösen. Wenn diese Ereignisse eintreten, müssen Sie bestimmte Rückrufmethoden als Ereignisprozeduren verwenden. Die NetStream-Klasse gibt die folgenden Metadatenereignisse an, die während der Wiedergabe eintreten können: `onCuePoint` (nur FLV-Dateien), `onImageData`, `onMetaData`, `onPlayStatus`, `onTextData` und `onXMPData`.

Sie müssen jedoch Rückrufmethoden für diese Prozeduren schreiben, da die Flash-Laufzeitumgebung andernfalls Fehler auslösen könnte. Beispielsweise wird mit dem folgenden Code eine FLV-Datei namens „video.flv“ aus dem gleichen Ordner wie Ihre SWF-Datei wiedergegeben:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);
ns.play("video.flv");
function asyncErrorHandler(event:AsyncErrorEvent):void
{
    trace(event.text);
}

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Mit dem oben stehenden Code wird eine lokale Videodatei mit dem Namen „video.flv“ geladen und überwacht, ob das `asyncError`-Ereignis (`AsyncErrorEvent.ASYNC_ERROR`) ausgelöst wird. Dieses Ereignis wird ausgelöst, wenn eine Ausnahme vom nativen asynchronen Code ausgelöst wird. In diesem Fall wird es ausgelöst, wenn die Videodatei Metadaten oder Cue-Point-Informationen enthält und die entsprechenden Listener nicht definiert wurden. Mit dem vorangegangenen Code wird das `asyncError`-Ereignis verarbeitet und der Fehler ignoriert, wenn Sie nicht an den Metadaten oder Cue-Point-Informationen in der Videodatei interessiert sind. Wenn Sie eine FLV-Datei mit Metadaten und verschiedenen Cue-Points haben, gibt die `trace()`-Funktion die folgenden Fehlermeldungen zurück:

```
Error #2095: flash.net.NetStream was unable to invoke callback onMetaData.
Error #2095: flash.net.NetStream was unable to invoke callback onCuePoint.
Error #2095: flash.net.NetStream was unable to invoke callback onCuePoint.
Error #2095: flash.net.NetStream was unable to invoke callback onCuePoint.
```

Die Fehler treten auf, da das NetStream-Objekt keine `onMetaData`- oder `onCuePoint`-Rückrufmethode finden konnte. Diese Rückrufmethoden können in Ihrer Anwendung auf verschiedene Arten definiert werden.

Verwandte Hilfethemen

Flash Media Server: Verarbeiten von Metadaten in Streams

Festlegen eines Objekts für die client-Eigenschaft des NetStream-Objekts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Durch Festlegen der `client`-Eigenschaft auf ein Objekt oder eine Unterklasse von NetStream können Sie die Rückrufmethoden `onMetaData` und `onCuePoint` entweder umleiten oder vollständig ignorieren. Im folgenden Code wird veranschaulicht, wie Sie ein leeres Objekt verwenden, um die Rückrufmethoden zu ignorieren, ohne das `asyncError`-Ereignis zu überwachen:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var customClient:Object = new Object();

var ns:NetStream = new NetStream(nc);
ns.client = customClient;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Wenn Sie die Rückrufmethode `onMetaData` oder `onCuePoint` überwachen möchten, müssen Sie Methoden zur Verarbeitung dieser Rückrufmethoden definieren, wie im folgenden Codeausschnitt dargestellt:

```
var customClient:Object = new Object();
customClient.onMetaData = metaDataHandler;
function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
```

Mit dem vorangegangenen Code wird die Rückrufmethode `onMetaData` überwacht und die `metaDataHandler()`-Methode aufgerufen, die einen String ausgibt. Wenn in der Flash-Laufzeitumgebung ein Cue-Point gefunden wird, werden keine Fehler ausgelöst, auch wenn keine `onCuePoint`-Rückrufmethode definiert ist.

Erstellen einer benutzerdefinierten Klasse und Definieren von Methoden zur Verarbeitung der Rückrufmethoden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem folgenden Code wird die `client`-Eigenschaft des NetStream-Objekts auf eine benutzerdefinierte Klasse eingestellt, die Prozeduren für die Rückrufmethoden definiert:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = new CustomClient();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Die CustomClient-Klasse sieht wie folgt aus:

```
package
{
    public class CustomClient
    {
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
    }
}
```

Die CustomClient-Klasse definiert eine Prozedur für die `onMetaData`-Rückrufprozedur. Wenn ein Cue-Point gefunden und die `onCuePoint`-Rückrufprozedur aufgerufen wurde, wird ein `asyncError`-Ereignis (`AsyncErrorEvent.ASYNC_ERROR`) mit der Meldung „flash.net.NetStream konnte Rückruf onCuePoint nicht auslösen“ ausgelöst. Um diesen Fehler zu vermeiden, müssen Sie entweder eine `onCuePoint`-Rückrufmethode in der CustomClient-Klasse oder eine Ereignisprozedur für das `asyncError`-Ereignis definieren.

Erweitern der NetStream-Klasse und Hinzufügen von Methoden zur Verarbeitung der Rückrufmethoden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem folgenden Code wird eine Instanz der CustomNetStream-Klasse erstellt, die in einem späteren Codebeispiel definiert wird:

```
var ns:CustomNetStream = new CustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Mit dem folgenden Code wird die CustomNetStream-Klasse definiert, die die NetStream-Klasse erweitert, das erforderliche NetConnection-Objekt erstellt und die Methoden der Rückrufprozeduren `onMetaData` und `onCuePoint` verarbeitet:

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    public class CustomNetStream extends NetStream
    {
        private var nc:NetConnection;
        public function CustomNetStream()
        {
            nc = new NetConnection();
            nc.connect(null);
            super(nc);
        }
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
        public function onCuePoint(infoObject:Object):void
        {
            trace("cue point");
        }
    }
}
```

Wenn Sie die Methoden `onMetaData()` und `onCuePoint()` in der CustomNetStream-Klasse umbenennen möchten, können Sie den folgenden Code verwenden:

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    public class CustomNetStream extends NetStream
    {
        private var nc:NetConnection;
        public var onMetaData:Function;
        public var onCuePoint:Function;
        public function CustomNetStream()
        {
            onMetaData = metaDataHandler;
            onCuePoint = cuePointHandler;
            nc = new NetConnection();
            nc.connect(null);
            super(nc);
        }
        private function metaDataHandler(infoObject:Object):void
        {
            trace("metadata");
        }
        private function cuePointHandler(infoObject:Object):void
        {
            trace("cue point");
        }
    }
}
```

Erweitern der NetStream-Klasse und Definieren als dynamische Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die NetStream-Klasse erweitern und diese Unterklasse als dynamische Klasse definieren, sodass die Rückrufmethoden `onCuePoint` und `onMetaData` dynamisch hinzugefügt werden können. Dies wird im folgenden Code veranschaulicht:

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Die DynamicCustomNetStream-Klasse sieht wie folgt aus:

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    public dynamic class DynamicCustomNetStream extends NetStream
    {
        private var nc:NetConnection;
        public function DynamicCustomNetStream()
        {
            nc = new NetConnection();
            nc.connect(null);
            super(nc);
        }
    }
}
```

Auch wenn keine Prozeduren für die Rückrufprozeduren `onMetaData` und `onCuePoint` vorhanden sind, werden keine Fehlermeldungen ausgelöst, da die DynamicCustomNetStream-Klasse dynamisch ist. Wenn Sie Methoden für die Rückrufmethoden `onMetaData` und `onCuePoint` definieren möchten, können Sie den folgenden Code verwenden:

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.onMetaData = metaDataHandler;
ns.onCuePoint = cuePointHandler;
ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
function cuePointHandler(infoObject:Object):void
{
    trace("cue point");
}
```

Setzen der client-Eigenschaft des NetStream-Objekts auf „this“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Durch Setzen der `client`-Eigenschaft auf `this` wird in der Anwendung im aktuellen Gültigkeitsbereich nach den Methoden `onMetaData()` und `onCuePoint()` gesucht. Dies wird im folgenden Codebeispiel gezeigt:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = this;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Wenn die Rückrufprozeduren `onMetaData` oder `onCuePoint` aufgerufen werden und keine Methoden zur Verarbeitung des Rückrufs vorhanden sind, werden keine Fehlermeldungen erzeugt. Um diese Rückrufprozeduren zu verarbeiten, erstellen Sie eine `onMetaData()`-Methode und eine `onCuePoint()`-Methode im Code, wie im folgenden Codeausschnitt dargestellt:

```
function onMetaData(infoObject:Object):void
{
    trace("metadata");
}
function onCuePoint(infoObject:Object):void
{
    trace("cue point");
}
```

Verwenden von Cue-Points und Metadaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den NetStream-Rückrufmethoden können Sie während der Videowiedergabe Cue-Point- und Metadatenereignisse erfassen und verarbeiten.

Verwenden von Cue-Points

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der folgenden Tabelle werden die Rückrufmethoden beschrieben, mit denen Sie in Flash Player und AIR F4V- und FLV-Cue-Points erfassen können.

Laufzeit	F4V	FLV
Flash Player 9/ AIR1.0		OnCuePoint
		OnMetaData
Flash Player 10		OnCuePoint
	OnMetaData	OnMetaData
	OnXMPData	OnXMPData

Im folgenden Beispiel wird eine einfache `for..in`-Schleife verwendet, um jede Eigenschaft im `infoObject`-Parameter, der von der `onCuePoint()`-Funktion empfangen wird, zu durchlaufen. Die trace()-Funktion wird aufgerufen, um eine Meldung anzuzeigen, wenn die Cue-Point-Daten empfangen werden:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = this;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

function onCuePoint(infoObject:Object):void
{
    var key:String;
    for (key in infoObject)
    {
        trace(key + ": " + infoObject[key]);
    }
}
```

Die folgende Ausgabe wird angezeigt:

```
parameters:
name: point1
time: 0.418
type: navigation
```

In diesem Code wird eines von mehreren möglichen Verfahren zum Festlegen des Objekts verwendet, für das die Rückrufmethode ausgeführt wird. Sie können auch andere Techniken verwenden; weitere Informationen finden Sie unter „Schreiben von Rückrufmethoden für Metadaten und Cue-Points“ auf Seite 518.

Verwenden von Video-Metadaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der `OnMetaData()`- und `OnXMPData()`-Funktionen können Sie auf die Metadateninformationen in der Videodatei, einschließlich von Cue-Points, zugreifen.

Verwenden von OnMetaData()

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Metadaten enthalten Angaben zur Videodatei wie Wiedergabedauer, Breite, Höhe und Bildrate. Welche Metadateninformationen Ihrer Videodatei hinzugefügt werden, ist von der Software abhängig, mit der Sie die Videodatei kodiert haben.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = this;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

function onMetaData(infoObject:Object):void
{
    var key:String;
    for (key in infoObject)
    {
        trace(key + ": " + infoObject[key]);
    }
}
```

Mit dem oben stehenden Code wird folgende Ausgabe generiert:

```
width: 320
audiodelay: 0.038
canSeekToEnd: true
height: 213
cuePoints: ,,
audiodatarate: 96
duration: 16.334
videodatarate: 400
framerate: 15
videocodecid: 4
audiocodecid: 2
```

Wenn das Video keine Audiodaten enthält, geben die audiobezogenen Metadateninformationen (z. B. `audiodatarate`) den Wert `undefined` zurück, da den Metadaten beim Kodieren keine Audioinformationen hinzugefügt werden.

Im oben stehenden Code wurden die Cue-Point-Informationen nicht angezeigt. Um die Cue-Point-Metadaten anzuzeigen, können Sie die folgenden Funktion verwenden, mit der die Elemente in einem Objekt rekursiv angezeigt werden:

```
function traceObject(obj:Object, indent:uint = 0):void
{
    var indentString:String = "";
    var i:uint;
    var prop:String;
    var val:*;
    for (i = 0; i < indent; i++)
    {
        indentString += "\t";
    }
    for (prop in obj)
    {
        val = obj[prop];
        if (typeof(val) == "object")
        {
            trace(indentString + " " + prop + ": [Object]");
            traceObject(val, indent + 1);
        }
        else
        {
            trace(indentString + " " + prop + ": " + val);
        }
    }
}
```

Mithilfe des vorangegangenen Codeausschnitts zum Senden des `infoObject`-Parameters in der `onMetaData()`-Methode wird die folgende Ausgabe erstellt:

```
width: 320
audiodatarate: 96
audiocodecid: 2
videocodecid: 4
videodatarate: 400
canSeekToEnd: true
duration: 16.334
audiodelay: 0.038
height: 213
framerate: 15
cuePoints: [Object]
    0: [Object]
        parameters: [Object]
            lights: beginning
        name: point1
        time: 0.418
        type: navigation
    1: [Object]
        parameters: [Object]
            lights: middle
        name: point2
        time: 7.748
        type: navigation
    2: [Object]
        parameters: [Object]
            lights: end
        name: point3
        time: 16.02
        type: navigation
```

Im folgenden Beispiel werden die Metadaten für ein MP4-Video angezeigt. Es wird vorausgesetzt, dass ein TextArea-Objekt namens `metaDataOut` vorhanden ist, in das die Metadaten geschrieben werden.

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.events.NetStatusEvent;
    import flash.media.Video;
    import flash.display.StageDisplayState;
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.MouseEvent;

    public class onMetaDataExample extends Sprite
    {
        var video:Video = new Video();

        public function onMetaDataExample():void
        {
            var videoConnection:NetConnection = new NetConnection();
            videoConnection.connect(null);

            var videoStream:NetStream = new NetStream(videoConnection);
            videoStream.client = this;

            addChild(video);
            video.x = 185;
            video.y = 5;

            video.attachNetStream(videoStream);

            videoStream.play("video.mp4");

            videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
        }

        public function onMetaData(infoObject:Object):void
        {
            for(var propName:String in infoObject)
            {
                metaDataOut.appendText(propName + "=" + infoObject[propName] + "\n");
            }
        }

        private function netStatusHandler(event:NetStatusEvent):void
        {
            if(event.info.code == "NetStream.Play.Stop")
                stage.displayState = StageDisplayState.NORMAL;
        }
    }
}
```

Die `onMetaData()`-Funktion hat für dieses Video die folgende Ausgabe erzeugt:

```
moovposition=731965
height=352
avclevel=21
videocodecid=avc1
duration=2.36
width=704
videoframerate=25
avcprofile=88
trackinfo=[object Object]
```

Verwenden des Informationsobjekts

In der folgenden Tabelle sind die möglichen Werte für Video-Metadaten aufgeführt, die an die `onMetaData()`-Rückruffunktion im von ihnen empfangenen Objekt übergeben werden:

Parameter	Beschreibung
aacaot	AAC-Audio-Objekttyp; 0, 1, oder 2 werden unterstützt.
avclevel	AVC IDC-Levelnummer, zum Beispiel 10, 11, 20, 21 usw.
avcprofile	AVC-Profilnummer, zum Beispiel 55, 77, 100 usw.
audiocodecid	Ein String, der angibt, welcher Audio-Codec (Kodier-/Dekodiertechnik) verwendet wurde, zum Beispiel „.Mp3“ oder „mp4a“.
audiodatarate	Eine Zahl, die die Abtastrate angibt, mit der das Audio kodiert wurde, in Kilobyte pro Sekunde.
audiodelay	Eine Zahl, die angibt, bei welcher Zeit in der FLV-Datei der Zeitpunkt 0 der ursprünglichen FLV-Datei vorhanden ist. Der Videoinhalt muss leicht verzögert werden, um korrekt mit dem Audio synchronisiert zu werden.
canSeekToEnd	Ein boolescher Wert. Die Einstellung `true` bedeutet, dass die FLV-Datei mit einem Schlüsselbild im letzten Bild kodiert wurde, das den Suchlauf bis zum Ende einer progressiv heruntergeladenen Videodatei ermöglicht. Der Wert lautet `false`, wenn die FLV-Datei nicht mit einem Schlüsselbild im letzten Bild kodiert wurde.
cuePoints	Ein Array von Objekten, eines für jeden in die FLV-Datei eingebetteten Cue-Point. Der Wert ist nicht definiert, wenn die FLV-Datei keine Cue-Points enthält. Jedes Objekt hat die folgenden Eigenschaften: • `type`: ein String, der den Typ des Cue-Points als „navigation“ oder „event“ angibt. • `name`: ein String, der den Namen des Cue-Points angibt. • `time`: eine Zahl, die die Zeit des Cue-Points in Sekunden mit einer Genauigkeit von drei Dezimalstellen (Millisekunden) angibt. • `parameters`: ein optionales Objekt mit Name-Wert-Paaren, die vom Benutzer beim Erstellen des Cue-Points festgelegt wurden.
duration	Eine Zahl, die die Dauer der Videodatei in Sekunden angibt.
framerate	Eine Zahl, die die Bildrate der FLV-Datei angibt.
height	Eine Zahl, die die Höhe der FLV-Datei in Pixel angibt.
seekpoints	Ein Array, das die verfügbaren Schlüsselbilder als Zeitstempel in Millisekunden auflistet. Optional.
tags	Ein Array von Schlüssel-Wert-Paaren, die die Informationen im „ilst“-Atom darstellen (entspricht bei MP4-Dateien den ID3-Tags). iTunes verwendet diese Tags. Kann zur Anzeige von Grafiken verwendet werden, falls verfügbar.
trackinfo	Objekt, das Informationen zu allen Tracks in der MP4-Datei enthält, einschließlich der Sample-Beschreibung-ID.

Parameter	Beschreibung
videocodecid	Ein String, der die Codec-Version angibt, mit der das Video kodiert wurde. Beispiel: „avc1“ oder „VP6F“
videodatarate	Eine Zahl, die die Video-Datenrate der FLV-Datei angibt.
videoframerate	Bildrate des MP4-Videos.
width	Eine Zahl, die die Breite der FLV-Datei in Pixel angibt.

In der folgenden Tabelle sind die möglichen Werte für den Parameter `videocodecid` aufgeführt:

videocodecid	Codec-Name
2	Sorenson H.263
3	Bildschirmvideo (nur SWF 7 und höher)
4	VP6 (nur SWF 8 und höher)
5	VP6-Video mit Alpha-Kanal (nur SWF 8 und höher)

In der folgenden Tabelle sind die möglichen Werte für den Parameter `audiocodecid` aufgeführt:

audiocodecid	Codec-Name
0	Unkomprimiert
1	ADPCM
2	MP3
4	Nellymoser bei 16 kHz Mono
5	Nellymoser, 8 kHz Mono
6	Nellymoser
10	AAC
11	Speex

Verwenden von onXMPData()

Flash Player 10 und höher, Adobe AIR 1.5 und höher

Die `onXMPData()`-Rückruffunktion empfängt spezifische Informationen der Adobe Extensible Metadata Platform (XMP), die in der Adobe F4V- oder FLV-Videodatei eingebettet sind. Die XMP-Metadaten enthalten sowohl Cue-Points als auch andere Video-Metadaten. XMP-Metadaten werden mit Flash Player 10 und Adobe 1.5 eingeführt und werden von späteren Versionen unterstützt.

Im folgenden Beispiel werden in den XMP-Metadaten enthaltene Cue-Point-Daten verarbeitet:

```
package
{
    import flash.display.*;
    import flash.net.*;
    import flash.events.NetStatusEvent;
    import flash.media.Video;

    public class onXMPDataExample extends Sprite
    {
        public function onXMPDataExample():void
        {
            var videoConnection:NetConnection = new NetConnection();
            videoConnection.connect(null);

            var videoStream:NetStream = new NetStream(videoConnection);
            videoStream.client = this;
            var video:Video = new Video();

            addChild(video);

            video.attachNetStream(videoStream);

            videoStream.play("video.f4v");
        }

        public function onMetaData(info:Object):void {
            trace("onMetaData fired");
        }

        public function onXMPData(infoObject:Object):void
        {
            trace("onXMPData Fired\n");
            //trace("raw XMP =\n");
            //trace(infoObject.data);
            var cuePoints:Array = new Array();
            var cuePoint:Object;
            var strFrameRate:String;
            var nTracksFrameRate:Number;
            var strTracks:String = "";
            var onXMPXML = new XML(infoObject.data);
            // Set up namespaces to make referencing easier
            var xmpDM:Namespace = new Namespace("http://ns.adobe.com/xmp/1.0/DynamicMedia/");
            var rdf:Namespace = new Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#");
            for each (var it:XML in onXMPXML..xmpDM::Tracks)
            {
```

```
                var strTrackName:String =
it.rdf::Bag.rdf::li.rdf::Description.@xmpDM::trackName;
                var strFrameRateXML:String =
it.rdf::Bag.rdf::li.rdf::Description.@xmpDM::frameRate;
                strFrameRate = strFrameRateXML.substr(1,strFrameRateXML.length);

                nTracksFrameRate = Number(strFrameRate);

                strTracks += it;
            }
            var onXMPTracksXML:XML = new XML(strTracks);
            var strCuepoints:String = "";
            for each (var item:XML in onXMPTracksXML..xmpDM::markers)
            {
                strCuepoints += item;
            }
            trace(strCuepoints);
        }
    }
}
```

Bei einer kurzen Videodatei namens „startrekintro.f4v“ werden im Beispiel die folgenden trace-Linien erzeugt. Die Linien zeigen die Cue-Point-Daten für Navigations- und Ereignis-Cue-Points in den XMP-Metadaten:

```
onMetaData fired
onXMPData Fired

<xmpDM:markers xmlns:xmp="http://ns.adobe.com/xap/1.0/"
xmlns:xmpDM="http://ns.adobe.com/xmp/1.0/DynamicMedia/"
xmlns:stDim="http://ns.adobe.com/xap/1.0/sType/Dimensions#"
xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:x="adobe:ns:meta/">
  <rdf:Seq>
    <rdf:li>
      <rdf:Description xmpDM:startTime="7695905817600" xmpDM:name="Title1"
xmpDM:type="FLVCuePoint" xmpDM:cuePointType="Navigation">
        <xmpDM:cuePointParams>
          <rdf:Seq>
            <rdf:li xmpDM:key="Title" xmpDM:value="Star Trek"/>
            <rdf:li xmpDM:key="Color" xmpDM:value="Blue"/>
          </rdf:Seq>
        </xmpDM:cuePointParams>
      </rdf:Description>
    </rdf:li>
    <rdf:li>
      <rdf:Description xmpDM:startTime="10289459980800" xmpDM:name="Title2"
xmpDM:type="FLVCuePoint" xmpDM:cuePointType="Event">
        <xmpDM:cuePointParams>
          <rdf:Seq>
            <rdf:li xmpDM:key="William Shatner" xmpDM:value="First Star"/>
            <rdf:li xmpDM:key="Color" xmpDM:value="Light Blue"/>
          </rdf:Seq>
        </xmpDM:cuePointParams>
      </rdf:Description>
    </rdf:li>
  </rdf:Seq>
</xmpDM:markers>
onMetaData fired
```

Hinweis: *In XMP-Daten wird die Zeit in Form von „DVA Ticks“ anstelle von Sekunden gespeichert. Sie berechnen die Cue-Point-Zeit, indem Sie die Startzeit durch die Bildrate teilen. Beispiel: Die Startzeit 7695905817600 geteilt durch die Bildrate 254016000000 ist gleich 30:30.*

Um die vollständigen unformatierten XMP-Metadaten, einschließlich der Bildrate, zu sehen, müssen Sie die Anmerkungsbezeichner (//) vor der zweiten und dritten `trace()`-Anweisung am Anfang der `onXMPData()`-Funktion entfernen.

Weitere Informationen zu XMP finden Sie unter:

- partners.adobe.com/public/developer/xmp/topic.html
- www.adobe.com/devnet/xmp/

Verwenden von Bildmetadaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das `onImageData`-Ereignisobjekt sendet Bilddaten als Byte-Array über einen AMF0-Datenkanal Die Daten können im Format JPEG, PNG oder GIF vorliegen. Definieren Sie eine `onImageData()`-Rückrufmethode zur Verarbeitung dieser Informationen wie Sie Rückrufmethoden für `onCuePoint` und `onMetaData` definieren würden. Im folgenden Beispiel wird eine `onImageData()`-Rückrufmethode verwendet, um auf Bilddaten zuzugreifen und diese anzuzeigen:

```
public function onImageData(imageData:Object):void
{
    // display track number
    trace(imageData.trackid);
    var loader:Loader = new Loader();
    //imageData.data is a ByteArray object
    loader.loadBytes(imageData.data);
    addChild(loader);
}
```

Verwenden von Textmetadaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das `onTextData`-Ereignis sendet Textdaten über einen AMF0-Datenkanal. Die Textdaten liegen im Format UTF-8 vor und enthalten zusätzliche Informationen über Formatierungen, die auf der „3GP Timed Text"-Spezifikation basieren. Diese Spezifikation definiert ein standardisiertes Format für Untertitel. Definieren Sie eine `onTextData()`-Rückrufmethode zur Verarbeitung dieser Informationen wie Sie Rückrufmethoden für `onCuePoint` oder `onMetaData` definieren würden. Im folgenden Beispiel wird die `onTextData()`-Methode verwendet, um die Track-ID und den entsprechenden Tracktext anzuzeigen.

```
public function onTextData(textData:Object):void
{
    // display the track number
    trace(textData.trackid);
    // displays the text, which can be a null string, indicating old text
    // that should be erased
    trace(textData.text);
}
```

Überwachen der NetStream-Aktivität

Flash Player 10.3 und höher, Adobe AIR 2.7 und höher

Sie können die NetStream-Aktivität überwachen und so die Informationen erfassen, die für eine Analyse der Mediennutzung sowie die Berichterstellung erforderlich sind. Die in diesem Abschnitt beschriebenen Überwachungsfunktionen ermöglichen die Erstellung von Bibliotheken mit Medienmessdaten. Die Daten werden dabei ohne enge Verknüpfung mit dem jeweiligen Videoplayer erfasst, in dem die Medien abgespielt werden. So können Client-Entwickler bei der Verwendung Ihrer Bibliothek ihre bevorzugten Videoplayer wählen. Verwenden Sie die NetMonitor-Klasse, um die Erstellung und Aktivität von NetStream-Objekten in einer Anwendung zu überwachen. Die NetMonitor-Klasse bietet eine Liste der jeweils aktiven NetStream-Objekte und löst bei jeder Erstellung eines NetStream-Objekts ein Ereignis aus.

Ein NetStream-Objekt löst die in der folgenden Tabelle aufgelisteten Ereignisse aus, je nach dem Typ der abgespielten Medien:

Ereignis	Progressiver Download	RTMP-Streaming	HTTP-Streaming
NetStream.Play.Start	Ja	Ja	Nein
NetStream.Play.Stop	Ja	Ja	Nein
NetStream.Play.Complete	Ja	Ja	Nein
NetStream.SeekStart.Notify	Ja	Ja	Ja
NetStream.Seek.Notify	Ja	Ja	Ja
NetStream.Unpause.Notify	Ja	Ja	Ja
NetStream.Unpause.Notify	Ja	Ja	Ja
NetStream.Play.Transition	–	Ja	–
NetStream.Play.TransitionComplete	–	Ja	–
NetStream.Buffer.Full	Ja	Ja	Ja
NetStream.Buffer.Flush	Ja	Ja	Ja
NetStream.Buffer.Empty	Ja	Ja	Ja

Das mit einer NetStream-Instanz verknüpfte NetStreamInfo-Objekt speichert auch die letzten Metadatenobjekte und XMP-Datenobjekte, die in den Medien erkannt wurden.

Beim Abspielen von Medien per HTTP-Streaming werden NetStream.Play.Start, NetStream.Play.Stop und NetStream.Play.Complete nicht ausgelöst, da die Anwendung vollständige Kontrolle über den Medienstream hat. Ein Videoplayer sollte diese Ereignisse für HTTP-Streams erstellen und auslösen.

Gleichfalls werden NetStream.Play.Transition und NetStream.Play.TransitionComplete weder für den progressiven Download noch für HTTP-Medien ausgelöst. Der dynamische Wechsel der Bitrate ist eine RTMP-Funktion. Wenn ein Videoplayer einen HTTP-Stream verwendet und eine ähnliche Funktion unterstützt, kann der Player transition-Ereignisse erstellen und auslösen.

Verwandte Hilfethemen

Adobe Developer Connection: Measuring video consumption in Flash

Überwachen von NetStream-Ereignissen

Zwei verschiedene Ereignistypen liefern wichtige Nutzungsdaten: `netStatus` und `mediaTypeData`. Außerdem kann ein Timer verwendet werden, um die Position des NetStream-Abspielkopfs regelmäßig aufzuzeichnen.

`netStatus`-Ereignisse stellen Informationen zur Verfügung, über die Sie feststellen können, welchen Teil eines Streams ein Benutzer betrachtet hat. Die transition-Ereignisse für Puffer- und RTMFP-Streams führen ebenfalls zu einem `netStatus`-Ereignis.

`mediaTypeData`-Ereignisse liefern Informationen zu Metadaten und XMP-Daten. Das Netstream.Play.Complete-Ereignis wird als `mediaTypeData`-Ereignis ausgelöst. Auch andere im Stream eingebettete Daten sind über `mediaTypeData`-Ereignisse verfügbar, wie Cue-Points, Text und Bilder.

Das folgende Beispiel veranschaulicht die Erstellung einer Klasse, die status- und data-Ereignisse von aktiven NetStream-Objekten in einer Anwendung überwacht. Normalerweise lädt eine solche Klasse die zu analysierenden Daten zur Erfassung auf einen Server.

```
package com.adobe.example
{
    import flash.events.NetDataEvent;
    import flash.events.NetMonitorEvent;
    import flash.events.NetStatusEvent;
    import flash.net.NetMonitor;
    import flash.net.NetStream;

    public class NetStreamEventMonitor
    {
        private var netmon:NetMonitor;
        private var heartbeat:Timer = new Timer( 5000 );

        public function NetStreamEventMonitor()
        {
            //Create NetMonitor object
            netmon = new NetMonitor();
            netmon.addEventListener( NetMonitorEvent.NET_STREAM_CREATE, newNetStream );

            //Start the heartbeat timer
            heartbeat.addEventListener( TimerEvent.TIMER, onHeartbeat );
            heartbeat.start();
        }

        //On new NetStream
        private function newNetStream( event:NetMonitorEvent ):void
        {
            trace( "New Netstream object");
            var stream:NetStream = event.netStream;
            stream.addEventListener(NetDataEvent.MEDIA_TYPE_DATA, onStreamData);
            stream.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
        }

        //On data events from a NetStream object
        private function onStreamData( event:NetDataEvent ):void
        {

            var netStream:NetStream = event.target as NetStream;
            trace( "Data event from " + netStream.info.uri + " at " + event.timestamp );
            switch( event.info.handler )
            {
                case "onMetaData":
                    //handle metadata;
                    break;
                case "onXMPData":
                    //handle XMP;
                    break;
                case "onPlayStatus":
                    //handle NetStream.Play.Complete
                case "onImageData":
                    //handle image
                    break;
                case "onTextData":
```

```
                //handle text
                break;
            default:
                //handle other events

        }
    }

    //On status events from a NetStream object
    private function onStatus( event:NetStatusEvent ):void
    {
        trace( "Status event from " + event.target.info.uri + " at " + event.target.time );
        //handle status events
    }
    //On heartbeat timer
    private function onHeartbeat( event:TimerEvent ):void
    {
        var streams:Vector.<NetStream> = netmon.listStreams();
        for( var i:int = 0; i < streams.length; i++ )
        {
            trace( "Heartbeat on " + streams[i].info.uri + " at " + streams[i].time );
            //handle heartbeat event
        }
    }

  }
}
```

Erkennen der Playerdomäne

Die URL und die Domäne der Webseite, auf der ein Benutzer Medieninhalte betrachtet, stehen nicht immer uneingeschränkt zur Verfügung. Sofern die Host-Website dies zulässt, können Sie die ExternalInterface-Klasse verwenden, um die genaue URL abzurufen. Auf einigen Websites, die die Verwendung von Videoplayern von Drittanbietern gestatten, ist die Nutzung von ExternalInterface jedoch nicht zulässig. In solchen Fällen können Sie die Domäne der aktuellen Webseite über die `pageDomain`-Eigenschaft der Security-Klasse ermitteln. Die vollständige URL wird nicht bekannt gemacht, um die Sicherheit der Benutzer sowie den Datenschutz zu gewährleisten.

Die Domäne der Seite kann über die statische `pageDomain`-Eigenschaft der Security-Klasse ermittelt werden:

```
var domain:String = Security.pageDomain;
```

Erweiterte Themen für Videodateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die folgenden Themen sprechen besondere Probleme beim Arbeiten mit FLV-Dateien an.

Konfigurieren von FLV-Dateien für das Hosten auf einem Server

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie mit FLV-Dateien arbeiten, müssen Sie Ihren Server gegebenenfalls für die Verwendung mit dem FLV-Dateiformat konfigurieren. MIME (Multipurpose Internet Mail Extensions) ist eine standardisierte Datenspezifikation, mit der Sie Dateien, die nicht aus ASCII-Zeichen bestehen, über Internetverbindungen übertragen können. Webbrowser und E-Mail-Clients sind für das Interpretieren zahlreicher MIME-Typen konfiguriert, sodass mit ihnen Video-, Audio- und Bilddaten sowie formatierte Texte gesendet und empfangen werden können. Um FLV-Dateien von einem Webserver laden zu können, müssen Sie eventuell die Dateierweiterung und den MIME-Typ auf dem Server registrieren. Sehen Sie hierzu die Dokumentation zu Ihrem Webserver ein. Der MIME-Typ für FLV-Dateien lautet `video/x-flv`. Die vollständigen Angaben für den FLV-Dateityp lauten:

- Mime-Typ: video/x-flv
- Dateierweiterung: .flv
- Erforderliche Parameter: keine
- Optionale Parameter: keine
- Bei der Kodierung zu berücksichtigen: FLV-Dateien sind Binärdateien. Bei einigen Anwendungen muss möglicherweise der Subtyp application/octet-stream eingerichtet werden.
- Sicherheitsprobleme: keine
- Veröffentlichte Spezifikation: www.adobe.com/go/video_file_format_de

Microsoft hat die Verwendung von Streaming-Media-Daten in Webserver Microsoft Internet Information Services (IIS) 6.0 im Vergleich zu den Vorversionen geändert. Bei älteren Versionen von IIS sind keine Änderungen erforderlich, um das Streaming von Flash Video-Dateien zu ermöglichen. In IIS6.0, dem Standardwebserver in Windows2003, benötigt der Server die Angabe eines MIME-Typs, um zu erkennen, dass es sich bei FLV-Dateien um Streaming-Media handelt.

Wenn SWF-Dateien für das Streaming externer FLV-Dateien unter Microsoft Windows Server® 2003 abgelegt und in einem Browser angezeigt werden, wird die SWF-Datei ordnungsgemäß wiedergegeben. Es erfolgt jedoch kein Streaming der FLV-Videodatei. Dieses Problem betrifft alle unter Windows Server 2003 abgelegten FLV-Dateien, auch Dateien, die mit früheren Versionen des Flash-Authoring-Tools erstellt werden, und das Macromedia Flash Video Kit für Dreamweaver MX 2004 von Adobe. Beim Testen unter anderen Betriebssystemen werden diese Dateien ordnungsgemäß ausgeführt.

Informationen zur Konfiguration von Microsoft Windows 2003 und Microsoft IIS Server 6.0 für das Streamen von FLV-Video finden Sie unter www.adobe.com/go/tn_19439_de.

Verwenden von Pfaden zu lokalen FLV-Dateien auf Macintosh-Computern

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie versuchen, eine lokale FLV-Datei auf einem Apple® Macintosh®-Computer von einem anderen als dem Systemlaufwerk wiederzugeben, indem Sie einen relativen Pfad mit einem Schrägstrich (/) angeben, wird das Video nicht wiedergegeben. Zu anderen Laufwerken als dem Systemlaufwerk zählen u. a. CD-ROM-Laufwerke, partitionierte Festplatten, Wechseldatenträger und angeschlossene Speichergeräte.

Hinweis: *Der Grund dafür liegt in einer Beschränkung des Betriebssystems und nicht an Flash Player oder AIR.*

Damit eine FLV-Datei auf einem Macintosh-Computer von einem anderen als dem Systemlaufwerk wiedergegeben wird, müssen Sie anstelle der relativen Schreibweise mit Schrägstrich (/) einen absoluten Pfad in der Schreibweise mit Doppelpunkt (:) verwenden. Im Folgenden ist der Unterschied zwischen beiden Schreibweisen aufgeführt:

- Schreibweise mit Schrägstrich**:** meinLaufwerk/meinOrdner/meineFLV.flv
- Schreibweise mit Doppelpunkt**:** (Mac OS®) meinLaufwerk/meinOrdner/meineFLV.flv

Video-Beispiel: Video Jukebox

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem folgenden Beispiel wird eine einfache Video-Jukebox erstellt, die dynamisch eine Liste von Videos lädt, um die darin enthaltenen Videos nacheinander wiederzugeben. Mit diesem Code können Sie eine Anwendung erstellen, die dem Benutzer z. B. das Blättern in einer Reihe von Video-Tutorials ermöglicht oder festlegt, welche Werbung angezeigt werden soll, bevor ein vom Benutzer angefordertes Video geliefert wird. In diesem Beispiel werden die folgenden Funktionen von ActionScript 3.0 veranschaulicht:

- Aktualisieren des Abspielkopfs basierend auf Wiedergabefortschritt einer Videodatei
- Überwachen und Einlesen der Metadaten einer Videodatei
- Verarbeiten von bestimmten Codes in einem Net-Stream
- Laden, Wiedergeben, Unterbrechen und Anhalten einer dynamisch geladenen FLV-Datei
- Ändern der Größe eines Video-Objekts in der Anzeigeliste basierend auf den Metadaten im Net-Stream

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „Video Jukebox“ befinden sich im Ordner „Samples/VideoJukebox“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
VideoJukebox.fla oder VideoJukebox.mxml	Die Hauptanwendungsdatei für Flex (MXML) oder Flash (FLA).
VideoJukebox.as	Die Klasse mit den Hauptfunktionen der Anwendung.
playlist.xml	Eine Datei, in der die Videodateien aufgeführt sind, die in die Video Jukebox geladen werden.

Laden einer externen Video-Playliste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die externe Datei „playlist.xml“ gibt die zu ladenden Videos und die Reihenfolge an, in der sie wiedergegeben werden. Zum Laden der XML-Datei benötigen Sie ein URLLoader-Objekt und ein URLRequest-Objekt. Dies wird im folgenden Code gezeigt:

```
uldr = new URLLoader();
uldr.addEventListener(Event.COMPLETE, xmlCompleteHandler);
uldr.load(new URLRequest(PLAYLIST_XML_URL));
```

Dieser Code wird im Konstruktor der VideoJukebox-Klasse eingefügt, sodass die Datei geladen wird, noch bevor anderer Code ausgeführt wird. Sobald die XML-Datei vollständig geladen ist, wird die `xmlCompleteHandler()`-Methode aufgerufen, die die externe Datei in ein XML-Objekt einliest, wie im folgenden Code dargestellt:

```
private function xmlCompleteHandler(event:Event):void
{
    playlist = XML(event.target.data);
    videosXML = playlist.video;
    main();
}
```

Das `playlist`-XML-Objekt enthält den unformatierten XML-Code aus der externen Datei, während videosXML ein XMLList-Objekt ist, das lediglich die Videoknoten enthält. Ein Beispiel für eine „playlist.xml"-Datei wird im folgenden Codeausschnitt gezeigt:

```
<videos>
    <video url="video/caption_video.flv" />
    <video url="video/cuepoints.flv" />
    <video url="video/water.flv" />
</videos>
```

Schließlich ruft die `xmlCompleteHandler()`-Methode die `main()`-Methode auf, mit der die verschiedenen Komponenteninstanzen in der Anzeigeliste sowie die NetConnection- und NetStream-Objekte festgelegt werden, die zum Laden der externen FLV-Dateien verwendet werden.

Erstellen der Benutzeroberfläche

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Erstellen der Benutzeroberfläche müssen Sie fünf Button-Instanzen in die Anzeigeliste ziehen und ihnen die folgenden Instanznamen zuweisen: `playButton`, `pauseButton`, `stopButton`, `backButton` und `forwardButton`.

Für jede dieser Button-Instanzen müssen Sie eine Prozedur für das `click`-Ereignis zuweisen, wie im folgenden Codeausschnitt dargestellt:

```
playButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
pauseButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
stopButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
backButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
forwardButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
```

Die `buttonClickHandler()`-Methode stellt mit einer switch-Anweisung fest, auf welche Schaltflächeninstanz geklickt wurde. Dies wird im folgenden Code gezeigt:

```
private function buttonClickHandler(event:MouseEvent):void
{
    switch (event.currentTarget)
    {
        case playButton:
            ns.resume();
            break;
        case pauseButton:
            ns.togglePause();
            break;
        case stopButton:
            ns.pause();
            ns.seek(0);
            break;
        case backButton:
            playPreviousVideo();
            break;
        case forwardButton:
            playNextVideo();
            break;
    }
}
```

Anschließend fügen Sie der Anzeigeliste eine Slider-Instanz hinzu und weisen ihr den Instanznamen `volumeSlider` zu. Mit dem folgenden Code wird die `liveDragging`-Eigenschaft der Slider-Instanz auf `true` gesetzt und ein Ereignis-Listener für das `change`-Ereignis der Slider-Instanz definiert:

```
volumeSlider.value = volumeTransform.volume;
volumeSlider.minimum = 0;
volumeSlider.maximum = 1;
volumeSlider.snapInterval = 0.1;
volumeSlider.tickInterval = volumeSlider.snapInterval;
volumeSlider.liveDragging = true;
volumeSlider.addEventListener(SliderEvent.CHANGE, volumeChangeHandler);
```

Fügen Sie der Anzeigeliste eine ProgressBar-Instanz hinzu und weisen Sie ihr den Instanznamen `positionBar` zu. Setzen Sie die zugehörige `mode`-Eigenschaft auf manual, wie im folgenden Code dargestellt:

```
positionBar.mode = ProgressBarMode.MANUAL;
```

Fügen Sie schließlich der Anzeigeliste eine Label-Instanz hinzu, und weisen Sie ihr den Instanznamen `positionLabel` zu. Der Wert dieser Label-Instanz wird mit der Timer-Instanz festgelegt.

Überwachen auf Metadaten eines Video-Objekts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn in Flash Player Metadaten für die einzelnen geladenen Videos gefunden werden, wird die Rückrufprozedur `onMetaData()` für die `client`-Eigenschaft des NetStream-Objekts aufgerufen. Mit dem folgenden Code wird ein Objekt initialisiert und die angegebene Rückrufprozedur eingerichtet:

```
client = new Object();
client.onMetaData = metadataHandler;
```

Die `metadataHandler()`-Methode kopiert ihre Daten in die meta-Eigenschaft, die bereits im Code definiert wurde. Auf diese Weise können Sie über die gesamte Anwendung auf die Metadaten des aktuellen Videos zugreifen. Als Nächstes wird das Video-Objekt auf der Bühne skaliert, sodass es den Abmessungen entspricht, die von den Metadaten zurückgegeben werden. Abschließend wird die Fortschrittleisten-Instanz positionBar basierend auf der Größe des aktuellen wiedergegebenen Videos verschoben und in der Größe geändert. Der folgende Code enthält die vollständige `metadataHandler()`-Methode:

```
private function metadataHandler(metadataObj:Object):void
{
    meta = metadataObj;
    vid.width = meta.width;
    vid.height = meta.height;
    positionBar.move(vid.x, vid.y + vid.height);
    positionBar.width = vid.width;
}
```

Dynamisches Laden von Video

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum dynamischen Laden von Video-Dateien verwendet die Anwendung ein NetConnection- und ein NetStream-Objekt. Mit dem folgenden Code wird ein NetConnection-Objekt erstellt und der Wert `null` an die `connect()`-Methode übergeben. Durch Angabe von `null` wird in Flash Player eine Verbindung mit einem Video auf dem lokalen Server anstelle einer Verbindung mit einem Remote-Server wie Flash Media Server hergestellt.

Mit dem folgenden Code wird die NetConnection- und die NetStream-Instanz erstellt, ein Ereignis-Listener für das `netStatus`-Ereignis definiert und der `client`-Eigenschaft das `client`-Objekt zugewiesen:

```
nc = new NetConnection();
nc.connect(null);

ns = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
ns.client = client;
```

Die `netStatusHandler()`-Methode wird immer aufgerufen, wenn sich der Videostatus ändert. Hierzu gehören das Starten oder Stoppen der Videowiedergabe, das Puffern von Videodaten oder wenn ein Video-Stream nicht gefunden werden kann. Im folgenden Code ist die `netStatusHandler()`-Ereignisprozedur aufgeführt:

```
private function netStatusHandler(event:NetStatusEvent):void
{
    try
    {
        switch (event.info.code)
        {
            case "NetStream.Play.Start":
                t.start();
                break;
            case "NetStream.Play.StreamNotFound":
            case "NetStream.Play.Stop":
                t.stop();
                playNextVideo();
                break;
        }
    }
    catch (error:TypeError)
    {
        // Ignore any errors.
    }
}
```

Im vorangegangenen Code wird die code-Eigenschaft des info-Objekts ausgewertet und ermittelt, ob der Code „NetStream.Play.Start“, „NetStream.Play.StreamNotFound“ oder „NetStream.Play.Stop“ enthält. Alle weiteren Codes werden ignoriert. Beim Starten des Net-Streams startet der Code die Timer-Instanz, die den Abspielkopf aktualisiert. Kann der Net-Stream nicht gefunden werden oder ist er angehalten, wird die Timer-Instanz gestoppt und die Anwendung versucht, das nächste Video in der Playliste wiederzugeben.

Jedes Mal, wenn der Timer ausgeführt wird, aktualisiert die Fortschrittleisten-Instanz `positionBar` die aktuelle Position, indem die `setProgress()`-Methode der ProgressBar-Klasse aufgerufen wird. Die Label-Instanz `positionLabel` wird mit der verstrichenen Zeit und der Gesamtlaufzeit des aktuellen Videos aktualisiert.

```
private function timerHandler(event:TimerEvent):void
{
    try
    {
        positionBar.setProgress(ns.time, meta.duration);
        positionLabel.text = ns.time.toFixed(1) + " of " meta.duration.toFixed(1) + " seconds";
    }
    catch (error:Error)
    {
        // Ignore this error.
    }
}
```

Steuern der Videolautstärke

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die Lautstärke eines dynamisch geladenen Videos steuern, indem Sie die `soundTransform`-Eigenschaft des NetStream-Objekts festlegen. Mit der Anwendung „Video Jukebox“ können Sie die Lautstärke einstellen, indem Sie den Wert der Slider-Instanz `volumeSlider` ändern. Im folgenden Code wird gezeigt, wie Sie die Lautstärke ändern können, indem Sie den Wert der Slider-Komponente einem SoundTransform-Objekt zuweisen, das als `soundTransform`-Eigenschaft des NetStream-Objekts festgelegt ist:

```
private function volumeChangeHandler(event:SliderEvent):void
{
    volumeTransform.volume = event.value;
    ns.soundTransform = volumeTransform;
}
```

Steuern der Videowiedergabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der Rest der Anwendung steuert die Videowiedergabe, wenn das Video das Ende des Video-Streams erreicht oder der Benutzer zum vorherigen oder nächsten Video springt.

Mit der folgenden Methode wird die Video-URL vom XMLList-Objekt für die aktuell ausgewählte Indexposition abgerufen:

```
private function getVideo():String
{
    return videosXML[idx].@url;
}
```

Die `playVideo()`-Methode ruft die `play()`-Methode des NetStream-Objekts auf, um das derzeit ausgewählte Video zu laden:

```
private function playVideo():void
{
    var url:String = getVideo();
    ns.play(url);
}
```

Die `playPreviousVideo()`-Methode verringert den aktuellen Videoindex, ruft die `playVideo()`-Methode auf, um eine neue Videodatei zu laden, und zeigt die Fortschrittleiste an:

```
private function playPreviousVideo():void
{
    if (idx > 0)
    {
        idx--;
        playVideo();
        positionBar.visible = true;
    }
}
```

Die letzte Methode `playNextVideo()` erhöht den Videoindex und ruft die `playVideo()`-Methode auf. Wenn das aktuelle Video das letzte Video in der Wiedergabeliste ist, wird die `clear()`-Methode für das Video-Objekt aufgerufen und die `visible`-Eigenschaft der Fortschrittleisten-Instanz auf `false` gesetzt:

```
private function playNextVideo():void
{
    if (idx < (videosXML.length() - 1))
    {
        idx++;
        playVideo();
        positionBar.visible = true;
    }
    else
    {
        idx++;
        vid.clear();
        positionBar.visible = false;
    }
}
```

Verwenden der StageVideo-Klasse für die hardwarebeschleunigte Darstellung

Flash Player 10.2 und höher

Flash Player optimiert die Videoleistung mittels der Hardwarebeschleunigung für die H.264-Dekodierung. Weitere Leistungssteigerungen können Sie erzielen, indem Sie die StageVideo-API verwenden. Mithilfe von Bühnenvideo kann Ihre Anwendung die hardwarebeschleunigte Darstellung nutzen.

Folgende Laufzeitumgebungen unterstützen die StageVideo-API:

- Flash Player 10.2 und höher

Hinweis: *Ab Flash Player 11.4/AIR 3.4 können Sie die Kameraeingabe mit StageVideo verwenden.*

Unter Erste Schritte mit Bühnenvideo können Sie Quellcode herunterladen und weitere Einzelheiten zum Bühnenvideo nachlesen.

Eine Kurzanleitung zu StageVideo finden Sie unter Arbeiten mit Bühnenvideo.

Hardwarebeschleunigung mit StageVideo

Die hardwarebeschleunigte Darstellung, die Videoskalierung, Farbkonvertierung und Blitting unterstützt, verstärkt die Leistungsvorteile der hardwarebeschleunigten Dekodierung. Auf Geräten mit GPU-Hardwarebeschleunigung können Sie ein flash.media.StageVideo-Objekt verwenden, um Video direkt über die Hardware des Geräts zu verarbeiten. Beim direkten Verarbeiten kann die CPU andere Aufgaben wahrnehmen, während das Video von der GPU verarbeitet wird. Im Gegensatz dazu verwendet die ältere Video-Klasse meist die Darstellung durch die Software. Das Darstellen per Software findet in der CPU statt und kann die Systemressourcen stark beanspruchen.

Derzeit bieten nur wenige Geräte vollständige Unterstützung für die GPU-Beschleunigung. Über die Bühnenvideo-Funktion können Anwendungen jedoch die verfügbare Hardwarebeschleunigung optimal nutzen.

Die Video-Klasse wird durch die StageVideo-Klasse nicht ersetzt. In Kombination bieten diese beiden Klassen stets eine optimale Videoanzeige, die auf die jeweiligen Ressourcen des Geräts abgestimmt ist. Anwendungen nutzen die Hardwarebeschleunigung, indem sie auf die entsprechenden Ereignisse warten und nach Bedarf zwischen StageVideo und Video wechseln.

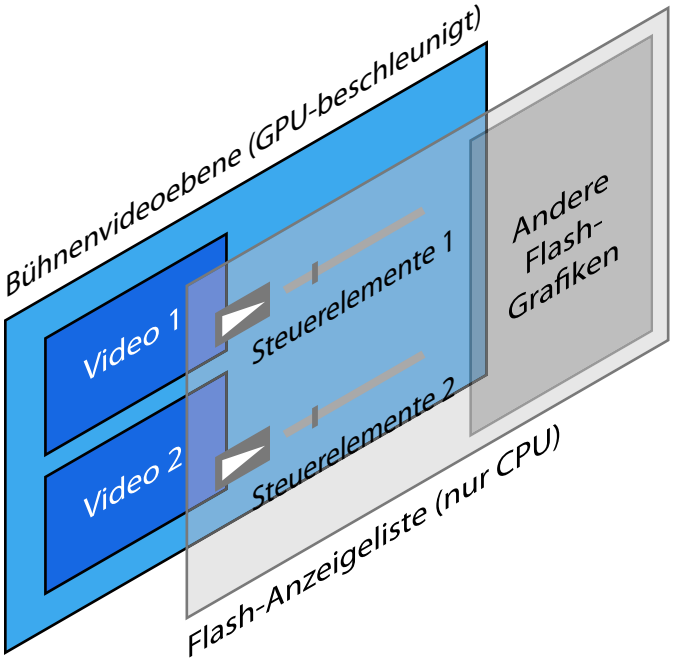
Bei Verwendung der StageVideo-Klasse gelten einige Einschränkungen für die Verwendung von Video. Bevor Sie die StageVideo-Funktion nutzen, sollten Sie diese Richtlinien lesen und sicherstellen, dass sie in Ihrer Anwendung hinnehmbar sind. Wenn die Einschränkungen akzeptabel sind, sollten Sie die StageVideo-Klasse immer dann verwenden, wenn Flash Player erkennt, dass die hardwarebeschleunigte Darstellung verfügbar ist. Einzelheiten finden Sie unter „Richtlinien und Einschränkungen“ auf Seite 547.

Parallele Ebenen: Bühnenvideo und die Flash-Anzeigeliste

Beim Bühnenvideomodell kann Flash Player das Video von der Anzeigeliste trennen. Flash Player teilt die zusammengesetzte Anzeige auf zwei Ebenen in Z-Anordnung auf.

Ebene für Bühnenvideo Die Ebene mit dem Bühnenvideo befindet sich im Hintergrund. Sie zeigt nur hardwarebeschleunigtes Video. Wegen dieser Architektur ist diese Ebene nicht verfügbar, wenn die Hardwarebeschleunigung auf dem Gerät nicht unterstützt wird oder nicht zur Verfügung steht. In ActionScript verarbeiten StageVideo-Objekte die Videos, die auf der Bühnenvideo-Ebene abgespielt werden.

Ebene für die Flash-Anzeigeliste Die Elemente der Flash-Anzeigeliste werden auf einer Ebene zusammengesetzt, die sich vor der Bühnenvideo-Ebene befindet. Zu den Elementen der Anzeigeliste gehören alle Komponenten, die von der Laufzeit gerendert werden, darunter auch Steuerelemente für die Wiedergabe. Wenn keine Hardwarebeschleunigung verfügbar ist, können Videos nur auf dieser Ebene abgespielt werden. Dazu wird ein Objekt der Video-Klasse verwendet. Das Bühnenvideo wird immer hinter den Grafiken der Flash-Anzeigeliste angezeigt.



Ebenen für die Videoanzeige

Das StageVideo-Objekt wird in einem rechteckigen Bereich des Bildschirms angezeigt, der am Fenster ausgerichtet und nicht gedreht ist. Objekte können nicht hinter der Bühnenvideo-Ebene angeordnet werden. Sie können jedoch die Ebene der Flash-Anzeigeliste verwenden, um andere Grafiken über der Bühnenvideo-Ebene anzuordnen. Bühnenvideo wird gleichzeitig mit der Anzeigeliste ausgeführt. So können Sie die beiden Mechanismen kombinieren, um einen einheitlichen visuellen Effekt zu erzielen, der auf zwei verschiedenen Ebenen beruht. Beispielsweise können Sie die vordere Ebene für Wiedergabesteuerelemente verwenden, mit denen das Bühnenvideo, das im Hintergrund ausgeführt wird, gesteuert werden kann.

Bühnenvideo und H.264-Codec

In Flash Player-Anwendungen sind zum Implementieren der Video-Hardwarebeschleunigung zwei Schritte erforderlich:

1 Kodieren des Videos im H.264-Format

2 Implementieren der StageVideo-API

Um beste Ergebnisse zu erzielen, sollten Sie beide Schritte ausführen. Über den H.264-Codec können Sie die Hardwarebeschleunigung optimal nutzen – von der Videodekodierung bis zur Darstellung.

Beim Bühnenvideo ist es nicht erforderlich, Daten von der GPU in die CPU zurückzulesen. Anders ausgedrückt: Die GPU sendet die dekodierten Bilder nicht an die CPU zurück, um sie mit den Objekten der Anzeigeliste zusammenzusetzen. Stattdessen blittet die GPU dekodierte und gerenderte Bilder direkt auf dem Bildschirm, hinter den Objekten der Anzeigeliste. Bei dieser Technik werden CPU und Arbeitsspeicher weniger stark beansprucht. Außerdem wird eine genauere Pixeldarstellung erzielt.

Verwandte Hilfethemen

„Videoformate“ auf Seite 506

Richtlinien und Einschränkungen

Wenn das Video im Vollbildmodus ausgeführt wird, ist das Bühnenvideo immer verfügbar, sofern das Gerät die Hardwarebeschleunigung unterstützt. Flash Player wird jedoch auch in einem Browser ausgeführt. Im Browserkontext wirkt sich die `wmode`-Einstellung auf die Verfügbarkeit des Bühnenvideos aus. Wenn Sie Bühnenvideo verwenden möchten, sollten Sie nach Möglichkeit immer die Einstellung `wmode="direct"` verwenden. Bühnenvideo ist nicht mit anderen `wmode`-Einstellungen kompatibel, wenn nicht der Vollbildmodus genutzt wird. Diese Einschränkung hat zur Folge, dass die Verfügbarkeit des Bühnenvideos zur Laufzeit unvorhersehbar schwanken kann. Wenn der Benutzer beispielsweise den Vollbildmodus beendet, während Bühnenvideo angezeigt wird, wechselt das Video zurück in den Browserkontext. Wenn der `wmode`-Parameter des Browsers nicht auf `"direct"` eingestellt ist, kann es vorkommen, dass das Bühnenvideo plötzlich nicht mehr verfügbar ist. Flash Player verwendet mehrere Ereignisse, um Anwendungen über Kontextänderungen während der Wiedergabe zu informieren. Wenn Sie die StageVideo-API implementieren, sollten Sie ein Video-Objekt als Reserve verwenden, für den Fall, dass das Bühnenvideo nicht verfügbar ist.

Wegen seiner direkten Beziehung mit der Hardware gelten beim Bühnenvideo einige Einschränkungen hinsichtlich der Videofunktionen. Die folgenden Einschränkungen müssen bei Bühnenvideo beachtet werden:

- Für jede SWF-Datei begrenzt Flash Player die Anzahl der StageVideo-Objekte, in denen gleichzeitig Video angezeigt werden kann, auf 4. Je nach den Hardwareressourcen des Geräts kann dieser Wert jedoch auch noch niedriger sein.
- Die Videozeitgebung ist nicht mit der Zeitgebung von Inhalten, die die Laufzeitumgebung anzeigt, synchronisiert.
- Der Videoanzeigebereich kann nur ein Rechteck sein. Kompliziertere Anzeigebereiche, zum Beispiel Ellipsen oder unregelmäßige Formen, können nicht verwendet werden.
- Sie können das Video nicht drehen.
- Die Bitmapzwischenspeicherung des Videos ist nicht möglich. Das BitmapData-Objekt kann nicht für den Zugriff auf das Video verwendet werden.
- Sie können keine Filter auf das Video anwenden.
- Sie können keine Farbtransformierungen auf das Video anwenden.
- Sie können keinen Alpha-Wert auf das Video anwenden.

- Mischmodi, die Sie auf Objekte auf der Anzeigenlistenebene anwenden, gelten nicht für Bühnenvideo.
- Sie können das Video nur auf vollen Pixelgrenzen platzieren.
- Obwohl das GPU-Rendern für die jeweilige Gerätehardware optimal ist, wird keine hundertprozentig identische Pixeldarstellung auf verschiedenen Geräten erzielt. Es treten geringfügige Abweichungen auf, die auf Unterschiede bei den Treibern und den Plattformen zurückzuführen sind.
- Einige Geräte unterstützen nicht alle erforderlichen Farbräume. So bieten manche Geräte beispielsweise keine Unterstützung für BT.709, den H.264-Standard. In diesen Fällen kann BT.601 zur schnellen Anzeige verwendet werden.
- Bühnenvideo kann nicht mit WMODE-Einstellungen wie „normal“, „opaque“ oder „transparent“ verwendet werden. Bühnenvideo unterstützt nur die Einstellung `WMODE=direct`, wenn es sich nicht im Vollbildmodus befindet. WMODE hat in Safari 4 oder höher und IE 9 oder höher keine Auswirkungen.

Videoplayer-Anwendungen sind in den meisten Fällen nicht von diesen Einschränkungen betroffen. Wenn diese Einschränkungen hinnehmbar sind, sollten Sie nach Möglichkeit immer Bühnenvideo verwenden.

Verwandte Hilfethemen

„Verwenden des Vollbildmodus“ auf Seite 178

Verwenden der StageVideo-APIs

Bühnenvideo ist ein Mechanismus in der Laufzeit, der die Videowiedergabe und die Geräteleistung optimiert. Die Laufzeit erstellt und verwaltet diesen Mechanismus – Ihre Aufgabe als Entwickler ist es, Ihre Anwendung so zu konfigurieren, dass sie den Mechanismus optimal nutzen kann.

Zur Verwendung von Bühnenvideo implementieren Sie verschiedene Ereignisprozeduren, die erkennen, wann Bühnenvideo verfügbar bzw. nicht verfügbar ist. Wenn Sie benachrichtigt werden, dass Bühnenvideo verfügbar ist, rufen Sie ein StageVideo-Objekt von der `Stage.stageVideos`-Eigenschaft ab. Die Laufzeit füllt dieses Vector-Objekt mit einem oder mehreren StageVideo-Objekten aus. Dann können Sie eines der StageVideo-Objekte anstelle eines Video-Objekts verwenden, um Streaming-Video anzuzeigen.

Wenn Sie in Flash Player benachrichtigt werden, dass das Bühnenvideo nicht mehr verfügbar ist, schalten Sie den Videostream zurück auf das Video-Objekt.

Hinweis: *Sie können keine StageVideo-Objekte erstellen.*

Stage.stageVideos-Eigenschaft

Die `Stage.stageVideos`-Eigenschaft ist ein Vector-Objekt, das Ihnen Zugriff auf StageVideo-Instanzen ermöglicht. Je nach Hardware- und Systemressourcen kann dieses Vector-Objekt maximal vier StageVideo-Objekte enthalten. Auf Mobilgeräten steht möglicherweise nur ein oder auch gar kein StageVideo-Objekt zur Verfügung.

Wenn kein Bühnenvideo verfügbar ist, enthält dieses Vector-Objekt keine Objekte. Um Laufzeitfehler zu vermeiden, darf der Zugriff auf Mitglieder dieses Vektors nur erfolgen, wenn das neueste `StageVideoAvailability`-Ereignis darauf hinweist, dass Bühnenvideo verfügbar ist.

StageVideo-Ereignisse

Die StageVideo-API bietet die folgenden Ereignisse:

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY Wird gesendet, wenn die `Stage.stageVideos`-Eigenschaft sich ändert. Die `StageVideoAvailabilityEvent.availability`-Eigenschaft hat den Wert `AVAILABLE` (VERFÜGBAR) oder `UNAVAILABLE` (NICHT VERFÜGBAR). Verwenden Sie dieses Ereignis, um zu ermitteln, ob die

`stageVideos`-Eigenschaft StageVideo-Objekte enthält, anstatt direkt die Länge des `Stage.stageVideos`-Vektors zu überprüfen.

StageVideoEvent.RENDER_STATE Wird gesendet, wenn ein NetStream- oder Camera-Objekt einem StageVideo-Objekt zugeordnet wurde und abgespielt wird. Gibt an, welches Dekodierverfahren gerade verwendet wird: Hardware, Software oder nicht verfügbar (nichts wird angezeigt). Das Ereignisziel enthält die `videoWidth`- und `videoHeight`-Eigenschaften, mit denen die Größe des Video-Viewports auf sichere Weise geändert werden kann.

Wichtig: *Die vom StageVideo-Zielobjekt abgerufenen Koordinaten stützen sich auf Bühnenkoordinaten, da sie nicht zur Standardanzeigeliste gehören.*

VideoEvent.RENDER_STATE Wird gesendet, wenn ein Video-Objekt verwendet wird. Das Ereignis gibt an, ob die Dekodierung über die Software oder über die Hardwarebeschleunigung durchgeführt wird. Wenn dieses Ereignis auf eine hardwarebeschleunigte Dekodierung hinweist, sollten Sie nach Möglichkeit zu einem StageVideo-Objekt wechseln. Das Video-Ereignisziel enthält die `videoWidth`- und `videoHeight`-Eigenschaften, mit denen die Größe des Video-Viewports auf sichere Weise geändert werden kann.

Arbeitsablauf zum Implementieren der StageVideo-Funktion

Das Implementieren der StageVideo-Funktion besteht aus den folgenden Hauptschritten:

1. Verwenden Sie einen Listener für das `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`-Ereignis, um festzustellen, wann sich der `Stage.stageVideos`-Vektor geändert hat. Siehe „Verwenden des StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY-Ereignisses“ auf Seite 550.
2. Wenn das `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`-Ereignis meldet, dass Bühnenvideo verfügbar ist, greifen Sie mit dem `Stage.stageVideos`-Vektorobjekt in dieser Ereignisprozedur auf ein StageVideo-Objekt zu.
3. Fügen Sie ein NetStream-Objekt unter Verwendung von `StageVideo.attachNetStream()` oder ein Camera-Objekt unter Verwendung von `StageVideo.attachCamera()` an.
4. Spielen Sie das Video mit `NetStream.play()` ab.
5. Verwenden Sie einen Listener für das `StageVideoEvent.RENDER_STATE`-Ereignis des StageVideo-Objekts, um den Wiedergabestatus des Videos festzustellen. Der Empfang dieses Ereignisses gibt auch an, dass die width- und height-Eigenschaften (Breite und Höhe) des Videos initialisiert oder geändert wurden. Siehe „Verwenden der StageVideoEvent.RENDER_STATE- und VideoEvent.RENDER_STATE-Ereignisse“ auf Seite 552.
6. Verwenden Sie einen Listener für das `VideoEvent.RENDER_STATE`-Ereignis des Video-Objekts. Dieses Ereignis stellt dieselben Statusangaben bereit wie `StageVideoEvent.RENDER_STATE`, sodass Sie damit auch feststellen können, ob die GPU-Beschleunigung verfügbar ist. Der Empfang dieses Ereignisses gibt auch an, dass die width- und height-Eigenschaften (Breite und Höhe) des Videos initialisiert oder geändert wurden. Siehe „Verwenden der StageVideoEvent.RENDER_STATE- und VideoEvent.RENDER_STATE-Ereignisse“ auf Seite 552.

Initialisieren von StageVideo-Ereignis-Listenern

Richten Sie die StageVideoAvailabilityEvent- und VideoEvent-Listener während der Anwendungsinitialisierung ein. Beispielsweise können Sie diese Listener in der `flash.events.Event.ADDED_TO_STAGE`-Ereignisprozedur initialisieren. Dieses Ereignis gewährleistet, dass die Anwendung auf der Bühne sichtbar ist:

```
public class SimpleStageVideo extends Sprite
    private var nc:NetConnection;
    private var ns:NetStream;

    public function SimpleStageVideo()
    {
        // Constructor for SimpleStageVideo class
        // Make sure the app is visible and stage available
        addEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
    }

    private function onAddedToStage(event:Event):void
    {
        //...
        // Connections
        nc = new NetConnection();
        nc.connect(null);
        ns = new NetStream(nc);
        ns.addEventListener(NetStatusEvent.NET_STATUS, onNetStatus);
        ns.client = this;

        // Screen
        video = new Video();
        video.smoothing = true;

        // Video Events
        // the StageVideoEvent.STAGE_VIDEO_STATE informs you whether
        // StageVideo is available
        stage.addEventListener(StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY,
            onStageVideoState);
        // in case of fallback to Video, listen to the VideoEvent.RENDER_STATE
        // event to handle resize properly and know about the acceleration mode running
        video.addEventListener(VideoEvent.RENDER_STATE, videoStateChange);
        //...
    }
```

Verwenden des StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY-Ereignisses

Geben Sie für die `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`-Prozedur ein Video- oder ein StageVideo-Objekt an, je nach der Verfügbarkeit von StageVideo. Wenn die `StageVideoAvailabilityEvent.availability`-Eigenschaft auf `StageVideoAvailability.AVAILABLE` gesetzt ist, verwenden Sie StageVideo. In diesem Fall können Sie sicher sein, dass der Stage.stageVideos-Vektor mindestens ein StageVideo-Objekt enthält. Rufen Sie ein StageVideo-Objekt von der `Stage.stageVideos`-Eigenschaft ab und weisen Sie ihm das NetStream-Objekt zu. Da StageVideo-Objekte immer im Hintergrund angezeigt werden, müssen Sie vorhandene Video-Objekte, die immer im Vordergrund erscheinen, entfernen. Fügen Sie mit dieser Ereignisprozedur auch einen Listener für das `StageVideoEvent.RENDER_STATE`-Ereignis hinzu.

Wenn die `StageVideoAvailabilityEvent.availability`-Eigenschaft auf `StageVideoAvailability.UNAVAILABLE` gesetzt ist, dürfen Sie weder StageVideo verwenden noch auf den `Stage.stageVideos`-Vektor zugreifen. Weisen Sie in diesem Fall das NetStream-Objekt einem Video-Objekt zu. Fügen Sie zum Schluss der Bühne das StageVideo- oder Video-Objekt hinzu und rufen Sie `NetStream.play()` auf.

Der folgende Code veranschaulicht die Verarbeitung des `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`-Ereignisses:

```
private var sv:StageVideo;
private var video:Video;

private function onStageVideoState(event:StageVideoAvailabilityEvent):void
{
    // Detect if StageVideo is available and decide what to do in toggleStageVideo
    toggleStageVideo(event.availability == StageVideoAvailability.AVAILABLE);
}

private function toggleStageVideo(on:Boolean):void
{
    // To choose StageVideo attach the NetStream to StageVideo
    if (on)
    {
        stageVideoInUse = true;
        if ( sv == null )
        {
            sv = stage.stageVideos[0];
            sv.addEventListener(StageVideoEvent.RENDER_STATE, stageVideoStateChange);
                sv.attachNetStream(ns);
        }

        if (classicVideoInUse)
        {
            // If you use StageVideo, remove from the display list the
            // Video object to avoid covering the StageVideo object
            // (which is always in the background)
            stage.removeChild ( video );
            classicVideoInUse = false;
        }
    } else
    {
        // Otherwise attach it to a Video object
        if (stageVideoInUse)
            stageVideoInUse = false;
        classicVideoInUse = true;
        video.attachNetStream(ns);
        stage.addChildAt(video, 0);
    }

    if ( !played )
    {
        played = true;
        ns.play(FILE_NAME);
    }
}
```

Wichtig: *Wenn eine Anwendung das erste Mal auf das Vektorelement bei Stage.stageVideos[0] zugreift, ist das Standardrechteck auf 0,0,0,0 eingestellt und für die Eigenschaften zum Verschieben und Ändern der Größe gelten die Standardwerte. Setzen Sie diese Werte immer auf Ihre bevorzugten Einstellungen zurück. Mit den* `videoWidth`*- und* `videoHeight`*-Eigenschaften des* `StageVideoEvent.RENDER_STATE`*- oder* `VideoEvent.RENDER_STATE`*-Ereignisziels können Sie die Abmessungen für den Video-Viewport berechnen.*

Sie können den vollständigen Quellcode dieser Beispielanwendung von der Website Erste Schritte mit Bühnenvideo herunterladen.

Verwenden der StageVideoEvent.RENDER_STATE- und VideoEvent.RENDER_STATE-Ereignisse

StageVideo- und Video-Objekte senden Ereignisse, um Anwendungen zu informieren, wenn die Anzeigeumgebung sich ändert. Diese Ereignisse heißen `StageVideoEvent.RENDER_STATE` und `VideoEvent.RENDER_STATE`.

Ein StageVideo- oder Video-Objekt löst ein Ereignis für den Renderstatus aus, wenn ein NetStream-Objekt zugewiesen ist und dessen Wiedergabe beginnt. Dieses Ereignis wird auch gesendet, wenn die Anzeigeumgebung sich ändert, beispielsweise wenn die Größe des Video-Viewports geändert wird. Verwenden Sie diese Benachrichtigungen, um den Viewport auf die aktuellen `videoHeight`- und `videoWidth`-Werte des Ereigniszielobjekts zurückzusetzen.

Für den Renderstatus können folgende Angaben gemeldet werden:

- `RENDER_STATUS_UNAVAILABLE`
- `RENDER_STATUS_SOFTWARE`
- `RENDER_STATUS_ACCELERATED`

Der Renderstatus gibt an, ob die hardwarebeschleunigte Dekodierung verwendet wird, unabhängig davon, welche Klasse derzeit Video abspielt. Überprüfen Sie anhand der `StageVideoEvent.status`-Eigenschaft, ob die erforderliche Dekodierung verfügbar ist. Wenn diese Eigenschaft auf „unavailable“ eingestellt ist, kann das StageVideo-Objekt das Video nicht abspielen. Bei diesem Status müssen Sie das NetStream-Objekt sofort wieder einem Video-Objekt zuweisen. Andere Statusangaben informieren Ihre Anwendung über die aktuellen Renderbedingungen.

Die folgende Tabelle veranschaulicht die Bedeutung aller Renderstatuswerte für StageVideoEvent- und VideoEvent-Objekte in Flash Player:

	VideoStatus.ACCELERATED	**VideoStatus.SOFTWARE**	**VideoStatus.UNAVAILABLE**
StageVideoEvent	Das Dekodieren und die Darstellung finden beide in der Hardware statt. (Optimal Leistung.)	Die Darstellung findet in der Hardware statt, das Dekodieren in der Software. (Akzeptable Leistung.)	Keine GPU-Ressourcen sind zum Verarbeiten des Videos verfügbar, es wird nichts angezeigt. **Verwenden Sie ein Video-Objekt als Ausweichlösung.**
VideoEvent	Die Darstellung findet in der Software statt, das Dekodieren in der Hardware. (Akzeptable Leistung nur auf modernen Desktopsystemen. Eingeschränkte Leistung im Vollbildmodus.)	Die Darstellung und das Dekodieren finden in der Software statt. (Schlechteste Leistung. Eingeschränkte Leistung im Vollbildmodus.)	–

Farbräume

Bühnenvideo unterstützt Farbräume über die zugrunde liegende Funktionalität der Hardware. SWF-Inhalt kann über Metadaten den bevorzugten Farbraum angeben. Ob dieser Farbraum auch verwendet werden kann, richtet sich jedoch nach der Grafikhardware des Geräts. Manche Geräte bieten Unterstützung für mehrere Farbräume, während andere Geräte gar keine Farbräume unterstützen. Wenn die Hardware den gewünschten Farbraum nicht unterstützt, versucht Flash Player, unter den unterstützten Farbräumen eine möglichst genaue Entsprechung zu finden.

Mit der `StageVideo.colorSpaces`-Eigenschaft können Sie feststellen, welche Farbräume von der Hardware unterstützt werden. Diese Eigenschaft gibt die Liste der unterstützten Farbräume in einem String-Vektor zurück:

```
var colorSpace:Vector.<String> = stageVideo.colorSpaces();
```

Anhand der `StageVideoEvent.colorSpace`-Eigenschaft können Sie ermitteln, welchen Farbraum das derzeit wiedergegebene Video verwendet. Überprüfen Sie diese Eigenschaft in der Ereignisprozedur für das `StageVideoEvent.RENDER_STATE`-Ereignis:

```
var currColorSpace:String;

//StageVideoEvent.RENDER_STATE event handler
private function stageVideoRenderState(event:Object):void
{
    //...
    currColorSpace = (event as StageVideoEvent).colorSpace;
    //...
}
```

Wenn Flash Player keinen Ersatz für einen nicht unterstützten Farbraum finden kann, verwendet das Bühnenvideo den standardmäßigen Farbraum, also BT.601. Videostreams mit der H.264-Kodierung verwenden beispielsweise in der Regel den Farbraum BT.709. Wenn die Gerätehardware den Farbraum BT.709 nicht unterstützt, gibt die `colorSpace`-Eigenschaft den Wert `"BT601"` zurück. Wenn `StageVideoEvent.colorSpace` den Wert `"unknown"` (unbekannt) zurückgibt, bietet die Hardware keine Möglichkeit, den Farbraum abzufragen.

Wenn der aktuelle Farbraum in Ihrer Anwendung als ungeeignet betrachtet wird, können Sie von einem StageVideo-Objekt zu einem Video-Objekt wechseln. Die Video-Klasse unterstützt alle Farbräume über das Software-Compositing.

Kapitel 26: Arbeiten mit Kameras

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine Kamera, die an einen Benutzercomputer angeschlossen ist, kann als Quelle für Videodaten dienen, die Sie mithilfe von ActionScript anzeigen und bearbeiten können. Die Camera-Klasse ist der ActionScript-Mechanismus für die Arbeit mit einer Computer- oder Gerätekamera.

Auf Mobilgeräten können Sie auch die CameraUI-Klasse verwenden. Die CameraUI-Klasse startet eine separate Kamera-Anwendung, mit der der Benutzer ein Bild oder Video aufnehmen kann. Nach der Aufnahme kann Ihre Anwendung über ein MediaPromise-Objekt auf das Bild oder das Video zugreifen.

Verwandte Hilfethemen

Christian Cantrell: How to use CameraUI in a Cross-platform Way

Michaël CHAIZE: Android, AIR and the Camera

Camera-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem Camera-Objekt können Sie eine Verbindung zu einer lokalen Kamera herstellen und das Video entweder lokal (zurück an den Benutzer) senden oder remote an einen Server (z. B. an Flash Media Server) übertragen.

Mithilfe der Camera-Klasse können Sie auf die folgenden Arten von Informationen zur Benutzerkamera zugreifen:

- Welche der auf dem Computer oder Gerät des Benutzers installierten Kameras stehen zur Verfügung
- Ist überhaupt eine Kamera installiert
- Ist Flash Player der Zugriff auf die Benutzerkamera gestattet oder nicht
- Welche Kameras sind derzeit aktiviert
- Wie lauten Größe und Höhe des erfassten Videos

Die Camera-Klasse umfasst mehrere nützliche Methoden und Eigenschaften für die Verwendung von Camera-Objekten. Beispielsweise enthält die statische `Camera.names`-Eigenschaft ein Array mit den Namen der Kameras, die derzeit auf dem Benutzercomputer installiert sind. Mit der `name`-Eigenschaft können Sie auch den Namen der aktuellen aktiven Kamera anzeigen.

Hinweis: *Beim Streaming von Kameravideo über das Netzwerk sollten Netzwerkunterbrechungen immer verarbeitet werden. Die Netzwerkverbindung kann aus zahlreichen Gründen unterbrochen werden, besonders auf Mobilgeräten.*

Anzeigen des Kamerainhalts auf dem Bildschirm

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für das Anschließen einer Kamera kann weniger Code erforderlich sein als für das Verwenden der NetConnection- und NetStream-Klasse zum Laden einer Videodatei. Andererseits kann die Camera-Klasse auch schnell kompliziert werden, da Sie mit Flash Player eine Genehmigung des Benutzers für eine Verbindung zu dessen Kamera benötigen, bevor Sie darauf zugreifen können.

Mit dem folgenden Code wird gezeigt, wie Sie mit der Camera-Klasse eine Verbindung zu einer lokal angeschlossenen Benutzerkamera herstellen können:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Hinweis: *Das Camera-Klasse hat keine Konstruktor-Methode. Um eine neue Camera-Instanz zu erstellen, müssen Sie die statische* `Camera.getCamera()`*-Methode verwenden.*

Entwerfen der Kamera-Anwendung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Schreiben einer Anwendung, die eine Verbindung mit einer Benutzerkamera herstellt, muss im Code Folgendes berücksichtigt werden:

- Prüfen, ob der Benutzer derzeit eine Kamera installiert hat. Verarbeiten einer Situation, in der keine Kamera verfügbar ist.
- Nur bei Verwendung von Flash Player müssen Sie prüfen, ob der Benutzer den Zugriff auf die Kamera ausdrücklich zugelassen hat. Aus Sicherheitsgründen wird das Dialogfeld „Flash Player-Einstellungen“ angezeigt, in dem Benutzer den Zugriff auf ihre Kamera zulassen oder verweigern können. Auf diese Weise wird verhindert, dass Flash Player eine Verbindung zu einer Benutzerkamera herstellt und einen Video-Stream ohne Genehmigung des Benutzers überträgt. Wenn der Benutzer auf „Zulassen“ klickt, kann Ihre Anwendung eine Verbindung zur Benutzerkamera herstellen. Wenn der Benutzer auf „Verweigern“ klickt, kann Ihre Anwendung keine Verbindung zur Benutzerkamera herstellen. Ihre Anwendungen sollten beide Fälle kulant verarbeiten.
- Überprüfen Sie in AIR, ob die Camera-Klasse für die von Ihrer Anwendung unterstützten Geräteprofile unterstützt wird.
- Die Camera-Klasse wird in mobilen Browsern nicht unterstützt.
- Die Camera-Klasse wird nicht in mobilen AIR-Anwendungen unterstützt, die den GPU-Rendermodus verwenden.
- Bei mobilen Geräten kann immer nur eine Kamera aktiv sein.

Verwandte Hilfethemen

Christophe Coenraets: Multi-User Video Tic-Tac-Toe

Mark Doherty: Android Radar app (Quelle)

Herstellen einer Verbindung mit der Kamera eines Benutzers

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der erste Schritt beim Herstellen einer Verbindung mit einer Kamera ist das Erstellen einer neuen Camera-Instanz. Dazu erstellen Sie zunächst eine Variable des Typs Camera und initialisieren diese dann mit dem Rückgabewert der statischen `Camera.getCamera()`-Methode.

Der nächste Schritt ist das Erstellen eines neuen Video-Objekts und das Anhängen des Camera-Objekts an das Video-Objekt.

Der dritte Schritt ist das Hinzufügen des Video-Objekts zur Anzeigeliste. Sie müssen die Schritte 2 und 3 ausführen, da die Camera-Klasse die DisplayObject-Klasse nicht erweitert und somit nicht direkt zur Anzeigeliste hinzugefügt werden kann. Um das von der Kamera erfasste Video anzuzeigen, erstellen Sie ein neues Video-Objekt, und rufen Sie die `attachCamera()`-Methode auf.

Diese drei Schritte werden im folgenden Beispielcode gezeigt:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Wenn ein Benutzer keine Kamera installiert hat, wird in der Anwendung keine Ausgabe angezeigt.

In der realen Welt müssen Sie zusätzliche Schritte für Ihre Anwendung ausführen. Weitere Informationen finden Sie unter „Prüfen auf installierte Kameras“ auf Seite 556 und „Erfassen der Zugriffsberechtigungen für eine Kamera“ auf Seite 557.

Verwandte Hilfethemen

Lee Brimelow: How to access the camera on Android devices

Prüfen auf installierte Kameras

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bevor Sie versuchen, Methoden oder Eigenschaften an einer Camera-Instanz anzuwenden, sollten Sie überprüfen, ob der Benutzer eine Kamera installiert hat. Es gibt zwei Möglichkeiten, um zu überprüfen, ob der Benutzer eine Kamera installiert hat:

- Prüfen Sie in der statischen `Camera.names`-Eigenschaft, welche Kameras verfügbar sind. Diese Eigenschaft enthält ein Array der Kameranamen. In der Regel enthält dieses Array maximal einen String, da die meisten Benutzer wahrscheinlich nur eine Kamera installiert haben. Mit dem folgenden Code wird gezeigt, wie Sie die Eigenschaft `Camera.names` überprüfen können, um festzustellen, ob der Benutzer verfügbare Kameras installiert hat:

```
if (Camera.names.length > 0)
{
    trace("User has at least one camera installed.");
    var cam:Camera = Camera.getCamera(); // Get default camera.
}
else
{
    trace("User has no cameras installed.");
}
```

- Überprüfen Sie den Rückgabewert der statischen `Camera.getCamera()`-Methode. Wenn keine Kameras verfügbar oder installiert sind, gibt diese Methode `null` zurück. Andernfalls gibt sie einen Verweis auf ein Camera-Objekt zurück. Im folgenden Code wird gezeigt, wie Sie mit der `Camera.getCamera()`-Methode überprüfen können, ob der Benutzer verfügbare Kameras installiert hat:

```
var cam:Camera = Camera.getCamera();
if (cam == null)
{
    trace("User has no cameras installed.");
}
else
{
    trace("User has at least 1 camera installed.");
}
```

Da die Camera-Klasse die DisplayObject-Klasse nicht erweitert, kann sie der Anzeigeliste nicht direkt mit der `addChild()`-Methode hinzugefügt werden. Um das von der Kamera erfasste Video anzuzeigen, müssen Sie ein neues Video-Objekt erstellen und die `attachCamera()`-Methode der Video-Instanz aufrufen.

Im folgenden Codeausschnitt wird gezeigt, wie Sie eine Verbindung mit einer vorhandenen Kamera herstellen. Wenn keine Kamera vorhanden ist, wird in der Anwendung keine Ausgabe angezeigt:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
```

Mobile Gerätekameras

Die Camera-Klasse wird nicht in der Flash Player-Laufzeit in mobilen Browsern unterstützt.

In AIR-Anwendungen auf Mobilgeräten können Sie auf die Kamera(s) im Gerät zugreifen. Bei mobilen Geräten können Sie sowohl die vordere als auch die hintere Kamera verwenden, aber es kann immer nur die Ausgabe einer Kamera angezeigt werden. (Beim Anschließen einer zweiten Kamera wird die erste entfernt.) Unter iOS wird die Kamera an der Vorderseite horizontal gespiegelt, unter Android dagegen nicht.

Erfassen der Zugriffsberechtigungen für eine Kamera

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der AIR-Anwendungs-Sandbox kann die Anwendung ohne Berechtigung des Benutzers auf alle Kameras zugreifen. Doch unter Android muss die Anwendung die Android-Berechtigung CAMERA im Anwendungsdeskriptor angeben.

Bevor die Ausgabe einer Kamera angezeigt werden kann, muss der Benutzer Flash Player den Zugriff auf die Kamera explizit gestatten. Wenn die `attachCamera()`-Methode aufgerufen wird, wird in Flash Player das Dialogfeld mit den Flash Player-Einstellungen angezeigt, in dem der Benutzer aufgefordert wird, den Zugriff auf die Kamera und das Mikrofon zuzulassen oder zu verweigern. Klickt der Benutzer auf die Schaltfläche „Zulassen“, zeigt Flash Player die Ausgabe der Kamera in der Video-Instanz auf der Bühne an. Klickt der Benutzer auf die Schaltfläche „Verweigern“, kann Flash Player keine Verbindung zur Kamera herstellen und das Video-Objekt bleibt leer.

Um festzustellen, ob der Benutzer Flash Player den Zugriff auf die Kamera erlaubt oder verweigert hat, können Sie das `status`-Ereignis der Kamera überwachen (`StatusEvent.STATUS`), wie im folgenden Code dargestellt:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    cam.addEventListener(StatusEvent.STATUS, statusHandler);
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
function statusHandler(event:StatusEvent):void
{
    // This event gets dispatched when the user clicks the "Allow" or "Deny"
    // button in the Flash Player Settings dialog box.
    trace(event.code); // "Camera.Muted" or "Camera.Unmuted"
}
```

Die `statusHandler()`-Funktion wird aufgerufen, sobald der Benutzer entweder auf „Zulassen“ oder „Verweigern“ klickt. Auf welche Schaltfläche der Benutzer geklickt hat, können Sie mit einer der folgenden beiden Methoden feststellen:

- Der `event`-Parameter der `statusHandler()`-Funktion enthält eine code-Eigenschaft mit dem String „Camera.Muted “oder „Camera.Unmuted“. Beim Wert „Camera.Muted“ hat der Benutzer auf die Schaltfläche „Verweigern“ geklickt und Flash Player kann nicht auf die Kamera zugreifen. Ein Beispiel hierfür wird im folgenden Codeausschnitt gezeigt:

  ```
  function statusHandler(event:StatusEvent):void
  {
      switch (event.code)
      {
          case "Camera.Muted":
              trace("User clicked Deny.");
              break;
          case "Camera.Unmuted":
              trace("User clicked Accept.");
              break;
      }
  }
  ```

- Die Camera-Klasse enthält eine schreibgeschützte Eigenschaft namens `muted`, mit der festgelegt wird, ob der Benutzer den Zugriff auf die Kamera im Flash Player-Bedienfeld „Zugriffsschutz“ verweigert (`true`) oder gewährt (`false`) hat. Ein Beispiel hierfür wird im folgenden Codeausschnitt gezeigt:

```
function statusHandler(event:StatusEvent):void
{
    if (cam.muted)
    {
        trace("User clicked Deny.");
    }
    else
    {
        trace("User clicked Accept.");
    }
}
```

Indem Sie überprüfen, ob das status-Ereignis ausgelöst wurde, können Sie Code schreiben, der das Zulassen oder Verweigern des Zugriffs auf die Kamera verarbeitet und entsprechend reagiert. Wenn der Benutzer auf die Schaltfläche „Verweigern“ klickt, können Sie den Benutzer in einer Meldung darauf hinweisen, dass er auf „Zulassen“ klicken muss, wenn er an einem Video-Chat teilnehmen möchte, oder Sie können sicherstellen, dass das Video-Objekt aus der Anzeigeliste gelöscht wird, um Systemressourcen freizugeben.

In AIR löst ein Camera-Objekt keine status-Ereignisse aus, da die Berechtigung zur Verwendung der Kamera nicht dynamisch ist.

Maximieren der Kamera-Videoqualität

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Standardmäßig sind neue Instanzen der Video-Klasse 320 Pixel breit und 240 Pixel hoch. Zur Maximierung der Videoqualität müssen Sie sicherstellen, dass Ihr Video-Objekt den Abmessungen des vom Camera-Objekt zurückgegebenen Videos entspricht. Sie können die Breite und Höhe des Camera-Objekts mithilfe der Eigenschaften `width` und `height` der Camera-Klasse abrufen. Sie können dann die Eigenschaften `width` und `height` des Video-Objekts so festlegen, dass sie den Abmessungen des Camera-Objekts entsprechen, oder Sie können die Breite und Höhe der Kamera an die Konstruktormethode der Video-Klasse übergeben. Dies wird im folgenden Codeausschnitt gezeigt:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video(cam.width, cam.height);
    vid.attachCamera(cam);
    addChild(vid);
}
```

Da die `getCamera()`-Methode einen Verweis auf ein Camera-Objekt zurückgibt (oder `null`, wenn keine Kameras verfügbar sind), können Sie auch dann auf die Methoden und Eigenschaften der Kamera zugreifen, wenn der Benutzer den Zugriff auf die Kamera verweigert hat. Auf diese Weise können Sie die Größe der Video-Instanz mithilfe der nativen Höhe und Breite der Kamera einstellen.

```
var vid:Video;
var cam:Camera = Camera.getCamera();

if (cam == null)
{
    trace("Unable to locate available cameras.");
}
else
{
    trace("Found camera: " + cam.name);
    cam.addEventListener(StatusEvent.STATUS, statusHandler);
    vid = new Video();
    vid.attachCamera(cam);
}
function statusHandler(event:StatusEvent):void
{
    if (cam.muted)
    {
        trace("Unable to connect to active camera.");
    }
    else
    {
        // Resize Video object to match camera settings and
        // add the video to the display list.
        vid.width = cam.width;
        vid.height = cam.height;
        addChild(vid);
    }
    // Remove the status event listener.
    cam.removeEventListener(StatusEvent.STATUS, statusHandler);
}
```

Informationen zum Vollbildmodus finden Sie unter „Festlegen der Stage-Eigenschaften“ auf Seite 176.

Überwachen des Kamerastatus

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Camera-Klasse enthält verschiedene Eigenschaften, mit denen Sie den aktuellen Status des Camera-Objekts überwachen können. Der folgende Code zeigt beispielsweise verschiedene Eigenschaften der Kamera an. Dazu werden ein Timer-Objekt und eine Textfeld-Instanz in der Anzeigeliste verwendet:

```
var vid:Video;
var cam:Camera = Camera.getCamera();
var tf:TextField = new TextField();
tf.x = 300;
tf.autoSize = TextFieldAutoSize.LEFT;
addChild(tf);

if (cam != null)
{
    cam.addEventListener(StatusEvent.STATUS, statusHandler);
    vid = new Video();
    vid.attachCamera(cam);
}
function statusHandler(event:StatusEvent):void
{
    if (!cam.muted)
    {
        vid.width = cam.width;
        vid.height = cam.height;
        addChild(vid);
        t.start();
    }
    cam.removeEventListener(StatusEvent.STATUS, statusHandler);
}

var t:Timer = new Timer(100);
t.addEventListener(TimerEvent.TIMER, timerHandler);
function timerHandler(event:TimerEvent):void
{
    tf.text = "";
    tf.appendText("activityLevel: " + cam.activityLevel + "\n");
    tf.appendText("bandwidth: " + cam.bandwidth + "\n");
    tf.appendText("currentFPS: " + cam.currentFPS + "\n");
    tf.appendText("fps: " + cam.fps + "\n");
    tf.appendText("keyFrameInterval: " + cam.keyFrameInterval + "\n");
    tf.appendText("loopback: " + cam.loopback + "\n");
    tf.appendText("motionLevel: " + cam.motionLevel + "\n");
    tf.appendText("motionTimeout: " + cam.motionTimeout + "\n");
    tf.appendText("quality: " + cam.quality + "\n");
}
```

Alle 100 Millisekunden wird das `timer`-Ereignis des Timer-Objekts ausgelöst, und die `timerHandler()`-Funktion aktualisiert das Textfeld in der Anzeigeliste.

Kapitel 27: Digitale Rechteverwaltung

Flash Player 10.1 und höher, Adobe AIR 1.5 und höher

Adobe® Access™ ist eine Lösung für den Inhaltsschutz. Mit dieser Lösung können Inhaltseigentümer, Distributoren und Werbetreibende neue Einnahmequellen erschließen, da sie einfachen Zugriff auf kostenpflichtige Inhalte gewähren können. Herausgeber verwenden Adobe Access zum Verschlüsseln von Inhalten, zum Erstellen von Richtlinien und zum Ausstellen von Lizenzen. Adobe Flash Player und Adobe AIR enthalten eine DRM-Bibliothek, das Adobe Access-Modul. Über dieses Modul kann die Laufzeitumgebung mit dem Adobe Access-Lizenzserver kommunizieren und geschützte Inhalte wiedergeben. Die Laufzeitumgebung schließt somit den Lebenszyklus von Inhalten ab, die durch Adobe Access geschützt und über Flash Media Server verteilt werden.

Mithilfe von Adobe Access können Inhaltsanbieter sowohl kostenlosen als auch kostenpflichtigen Inhalt bereitstellen. Angenommen, ein Verbraucher möchte ein Fernsehprogramm ohne Werbung sehen. Der Verbraucher registriert sich und zahlt eine Gebühr an den Inhaltsanbieter. Anschließend kann der Verbraucher seine Anmeldedaten eingeben, um das Programm abzurufen und ohne Werbepausen abzuspielen.

In einem anderen Beispiel möchte ein Verbraucher Inhalt offline anzeigen, während er ohne Internetverbindung auf Reisen ist. Dieser Offline-Arbeitsablauf wird in AIR-Anwendungen unterstützt. Nachdem der Benutzer sich registriert und die anfallende Gebühr entrichtet hat, kann er den Inhalt und die zugehörige AIR-Anwendung von der Website des Herausgebers herunterladen. Mithilfe der AIR-Anwendung kann der Benutzer den Inhalt während des zulässigen Zeitraums offline anzeigen. Der Inhalt kann auch mit anderen Geräten in derselben Gerätegruppe gemeinsam genutzt werden, wenn Domänen verwendet werden (**neu in 3.0**).

Adobe Access unterstützt auch den anonymen Zugriff, für den keine Benutzerauthentifizierung erforderlich ist. Beispielsweise kann ein Herausgeber über anonymen Zugriff mit Werbung unterlegten Inhalt bereitstellen oder den Zugriff auf den aktuellen Inhalt für einen bestimmten Zeitraum kostenlos gewähren. Der Anbieter kann auch den Typ und die Version der Programme, mit denen die Inhalte abgespielt werden können, einschränken.

Wenn ein Benutzer versucht, geschützten Inhalt in Flash Player oder Adobe AIR abzuspielen, muss die Anwendung die DRM-APIs aufrufen. Die DRM-APIs starten den Arbeitsablauf für die Wiedergabe des geschützten Inhalts. Die Laufzeit kontaktiert den Lizenzserver über das Adobe Access-Modul. Der Lizenzserver authentifiziert bei Bedarf den Benutzer und stellt eine Lizenz für die Wiedergabe des geschützten Inhalts aus. Die Laufzeit erhält die Lizenz und entschlüsselt den Inhalt zur Wiedergabe.

In diesem Kapitel wird beschrieben, wie Sie Ihre Anwendung so einrichten, dass durch Adobe Access geschützter Inhalt wiedergegeben werden kann. Sie müssen nicht wissen, wie Inhalt verschlüsselt wird und wie Richtlinien mit Adobe Access verwaltet werden. Es wird jedoch vorausgesetzt, dass Sie mit dem Adobe Access-Lizenzserver kommunizieren, um den Benutzer zu authentifizieren und die Lizenz abzurufen. Weiterhin wird davon ausgegangen, dass das Design Ihrer Anwendung das Abspielen von Inhalt, der durch Adobe Access geschützt ist, explizit ermöglicht.

Einen Überblick über Adobe Access und die Erstellung von Richtlinien finden Sie in der Dokumentation zu Adobe Access.

Verwandte Hilfethemen

flash.net.drm-Paket

flash.net.NetConnection

flash.net.NetStream

Arbeitsablauf für geschützte Inhalte

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Wichtig: In Flash Player 11.5 und höheren Versionen ist das Adobe Access-Modul bereits enthalten, sodass der Updateschritt (Aufrufen von `SystemUpdater.update(SystemUpdaterType.DRM)`) nicht erforderlich ist. Dies schließt die folgenden Browser und Plattformen ein:

- Flash Player 11.5 ActiveX-Steuerelement, für alle Plattformen außer Internet Explorer unter Windows 8 mit Intel-Prozessoren
- Flash Player 11.5-Plug-In, für alle Browser
- Adobe AIR (Desktop und Mobil)

Dies bedeutet, dass der Updateschritt in den folgenden Fällen *weiterhin erforderlich ist*:

- Internet Explorer unter Windows 8 mit Intel-Prozessoren
- Flash Player 11.4 und älter, außer mit Google Chrome 22 und höher (alle Plattformen) oder 21 und höher (Windows)

Hinweis: *Es treten keine Probleme auf, wenn Sie* `SystemUpdater.update(SystemUpdaterType.DRM)` *in einem System mit Flash Player 11.5 oder höher aufrufen, es wird jedoch nichts heruntergeladen.*

Im Folgenden wird in groben Zügen der Arbeitsablauf beschrieben, mit dem eine Anwendung geschützten Inhalt abrufen und wiedergeben kann. Bei diesem Arbeitsablauf wird davon ausgegangen, dass das Design der Anwendung das Abspielen von Inhalt, der durch Adobe Access geschützt wird, explizit ermöglicht.

1. Die Metadaten des Inhalts werden abgerufen.
2. Bei Bedarf wird Flash Player aktualisiert.
3. Es wird überprüft, ob eine Lizenz lokal verfügbar ist. Wenn ja, wird die Lizenz geladen und der Arbeitsablauf wird bei Schritt 7 fortgesetzt. Andernfalls wird der Arbeitsablauf bei Schritt 4 fortgesetzt.
4. Es wird überprüft, ob eine Authentifizierung erforderlich ist. Ist dies nicht der Fall, können Sie mit Schritt 7 fortfahren.
5. Wenn eine Authentifizierung erforderlich ist, werden die Authentifizierungsdaten vom Benutzer abgefragt und an den Lizenzserver übergeben.
6. Wenn die Domänenregistrierung erforderlich ist, treten Sie der Domäne bei (AIR 3.0 und höher).
7. Nach erfolgreicher Authentifizierung wird die Lizenz vom Server heruntergeladen.
8. Der Inhalt wird wiedergegeben.

Wenn kein Fehler auftritt und der Benutzer erfolgreich autorisiert wurde, die Inhalte anzuzeigen, löst das NetStream-Objekt ein DRMStatusEvent-Objekt aus. Dann beginnt die Anwendung mit der Wiedergabe. Das DRMStatusEvent-Objekt enthält die Gutscheininformationen, in denen die Richtlinien und Berechtigungen des Benutzers angezeigt werden. Beispielsweise definieren die Informationen in diesem Objekt, ob der Inhalt offline angezeigt werden kann und wie lange die Lizenz gültig ist. Die Anwendung kann diese Daten verwenden, um den Benutzer über den Status der Richtlinien zu informieren. Beispielsweise kann die Anwendung in einer Statusleiste die Anzahl der Tage anzeigen, die für die Ansicht der Inhalte verbleiben.

Wenn der Benutzer zum Offline-Zugriff berechtigt ist, wird der Gutschein zwischengespeichert und der verschlüsselte Inhalt wird auf das Gerät des Benutzers heruntergeladen. Der Inhalt ist so lange verfügbar, wie im Lizenz-Cache-Zeitraum angegeben. Die `detail`-Eigenschaft im Ereignis enthält die Angabe `DRM.voucherObtained`. Die Anwendung entscheidet, ob der Inhalt lokal gespeichert werden soll, damit er offline verfügbar ist. Sie können Gutscheine auch mit der DRMManager-Klasse vorausladen.

Hinweis: *Das Zwischenspeichern und Vorausladen von Gutscheinen wird sowohl in AIR als auch in Flash Player unterstützt. Das Herunterladen und Speichern von verschlüsselten Inhalten wird jedoch nur in AIR unterstützt.*

Die Anwendung ist dafür zuständig, die Fehlerereignisse explizit zu verarbeiten. Zu diesen Ereignissen zählen auch Fälle, in denen der Benutzer gültige Informationen eingegeben hat, der Gutschein, durch den die verschlüsselten Inhalte geschützt werden, jedoch den Zugriff auf die Inhalte einschränkt. Beispielsweise kann ein authentifizierter Benutzer nicht auf Inhalte zugreifen, wenn die anfallenden Gebühren nicht entrichtet wurden. Dieser Fall kann auch dann eintreten, wenn zwei Benutzer, die beide beim selben Herausgeber registriert sind, versuchen, Inhalte gemeinsam zu verwenden, für die nur ein Mitglied gezahlt hat. Die Anwendung muss den Benutzer über den Fehler informieren und eine Alternative anbieten. Eine typische Alternative umfasst Anleitungen dazu, wie der Benutzer sich registrieren und für die Anzeigeberechtigung zahlen kann.

Ausführlicher API-Arbeitsablauf

Flash Player 10.1 und höher, AIR 2.0 und höher

Im Folgenden wird der Arbeitsablauf für geschützte Inhalte ausführlicher beschrieben. Dabei werden auch die APIs vorgestellt, die für die Wiedergabe von durch Adobe Access geschützten Inhalten verwendet werden.

1. Verwenden Sie ein URLLoader-Objekt, um die Bytes der Metadaten-Datei des geschützten Inhalts zu laden. Stellen Sie dieses Objekt auf eine Variable ein, wie beispielsweise `metadata_bytes`.

   Alle von Adobe Access kontrollierten Inhalte verfügen über Adobe Access-Metadaten. Wenn der Inhalt verpackt wird, können diese Metadaten als separate Datei (.metadata) mit dem Inhalt gespeichert werden. Weitere Informationen finden Sie in der Dokumentation zu Adobe Access.

2. Erstellen Sie eine DRMContentData-Instanz. Fügen Sie diesen Code in einen try-catch-Block ein:

   ```
   new DRMContentData(metadata_bytes)
   ```

   Dabei ist *`metadata_bytes`* das URLLoader-Objekt aus Schritt 1.

3. (Nur Flash Player) Die Laufzeit sucht das Adobe Access-Modul. Falls es nicht gefunden wird, wird ein IllegalOperationError mit DRMErrorEvent-Fehlercode 3344 oder DRMErrorEvent-Fehlercode 3343 ausgegeben.

   Zur Verarbeitung dieses Fehlers laden Sie das Adobe Access-Modul über die SystemUpdater-API herunter. Nachdem dieses Modul heruntergeladen wurde, löst das SystemUpdater-Objekt ein COMPLETE-Ereignis aus. Fügen Sie einen Ereignis-Listener für dieses Ereignis ein, mit dem der Arbeitsablauf zu Schritt 2 zurückkehrt, wenn dieses Ereignis ausgelöst wird. Diese Schritte werden im folgenden Codebeispiel gezeigt:

   ```
   flash.system.SystemUpdater.addEventListener(Event.COMPLETE, updateCompleteHandler);
   flash.system.SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
   ```

   ```
   private function updateCompleteHandler (event:Event):void {
       /*redo step 2*/
       drmContentData = new DRMContentData(metadata_bytes);
   }
   ```

   Wenn der Player selbst aktualisiert werden muss, wird ein Statusereignis ausgelöst. Weitere Informationen zur Verarbeitung dieses Ereignisses finden Sie unter „Warten auf ein Aktualisierungsereignis“ auf Seite 580.

Hinweis: *In AIR-Anwendungen ist das AIR-Installationsprogramm für die Aktualisierung des Adobe Access-Moduls und die erforderlichen Laufzeitaktualisierungen zuständig.*

4 Erstellen Sie Listener, die auf die vom DRMManager-Objekt ausgelösten DRMStatusEvent- und DRMErrorEvent-Ereignisse warten:

```
DRMManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
DRMManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
```

Stellen Sie im DRMStatusEvent-Listener sicher, dass der Gutschein gültig (nicht null) ist. Verarbeiten Sie im DRMErrorEvent-Listener die DRMErrorEvents-Ereignisse. Einzelheiten finden Sie unter „Verwenden der DRMStatusEvent-Klasse“ auf Seite 572 und „Verwenden der DRMErrorEvent-Klasse“ auf Seite 577.

5 Laden Sie den Gutschein (Lizenz) zum Abspielen des Inhalts.

Versuchen Sie zunächst, eine lokal gespeicherte Lizenz zu laden:

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.LOCAL_ONLY)
```

Nach abgeschlossenem Ladevorgang löst das DRMManager-Objekt ein `DRMStatusEvent.DRM_Status`-Ereignis aus.

6 Wenn das DRMVoucher-Objekt nicht null ist, ist der Gutschein gültig. Fahren Sie mit Schritt 13 fort.

7 Wenn das DRMVoucher-Objekt null ist, überprüfen Sie die Authentifizierungsmethode, die die Richtlinie für diesen Inhalt vorgibt. Verwenden Sie die `DRMContentData.authenticationMethod`-Eigenschaft.

8 Wenn die Authentifizierungsmethode `ANONYMOUS` ist, fahren Sie mit Schritt 13 fort.

9 Bei der Authentifizierungsmethode `USERNAME_AND_PASSWORD` muss die Anwendung einen Mechanismus bereitstellen, über den der Benutzer seine Anmeldedaten eingeben kann. Übergeben Sie diese Anmeldedaten an den Lizenzserver, um den Benutzer zu authentifizieren:

```
DRMManager.authenticate(metadata.serverURL, metadata.domain, username, password)
```

Das DRMManager-Objekt löst ein `DRMAuthenticationErrorEvent`-Ereignis aus, wenn die Authentifizierung fehlschlägt, bzw. ein `DRMAuthenticationCompleteEvent`-Ereignis, wenn die Authentifizierung erfolgreich ist. Erstellen Sie Listener für diese Ereignisse.

10 Wenn die Authentifizierungsmethode `UNKNOWN` ist, muss eine benutzerdefinierte Authentifizierungsmethode verwendet werden. In diesem Fall hat der Content-Provider vorgesehen, dass die Authentifizierung auf eine Out-of-Band-Weise erfolgt, indem die ActionScript 3.0-APIs nicht verwendet werden. Das benutzerdefinierte Authentifizierungsverfahren muss ein Authentifizierungstoken erzeugen, das an die `DRMManager.setAuthenticationToken()`-Methode übergeben werden kann.

11 Wenn die Authentifizierung fehlschlägt, muss die Anwendung zu Schritt 9 zurückkehren. Ihre Anwendung sollte einen Mechanismus enthalten, der wiederholte Authentifizierungsfehler verarbeitet und einschränkt. Beispielsweise kann nach drei Versuchen eine Meldung eingeblendet werden, die dem Benutzer mitteilt, dass die Authentifizierung erfolglos war und der Inhalt deshalb nicht abgespielt werden kann.

12 Wenn Sie das gespeicherte Token verwenden möchten, anstatt den Benutzer zur Eingabe seiner Anmeldedaten aufzufordern, geben Sie das Token mit der `DRMManager.setAuthenticationToken()`-Methode an. Dann laden Sie die Lizenz vom Lizenzserver herunter und spielen den Inhalt ab, wie in Schritt 8.

13 (Optional) Wenn die Authentifizierung erfolgreich ist, können Sie das Authentifizierungstoken erfassen. Dies ist ein im Arbeitsspeicher zwischengespeichertes Byte-Array. Rufen Sie dieses Token mit der `DRMAuthenticationCompleteEvent.token`-Eigenschaft ab. Sie können das Authentifizierungstoken speichern, damit der Benutzer seine Anmeldedaten für diesen Inhalt nicht mehrfach eingeben muss. Der Lizenzserver bestimmt den Gültigkeitszeitraum des Authentifizierungstokens.

14 Bei erfolgreicher Authentifizierung laden Sie die Lizenz vom Lizenzserver herunter.

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.FORCE_REFRESH)
```

Nach abgeschlossenem Ladevorgang löst das DRMManager-Objekt ein DRMStatusEvent.DRM_STATUS-Ereignis aus. Warten Sie auf dieses Ereignis. Wenn es ausgelöst wird, kann der Inhalt wiedergegeben werden.

15 Spielen Sie das Video ab, indem Sie ein NetStream-Objekt erstellen und dann seine `play()`-Methode aufrufen:

```
stream = new NetStream(connection);
stream.addEventListener(DRMStatusEvent.DRM _STATUS, drmStatusHandler);
stream.addEventListener(DRMErrorEvent.DRM_ERROR, drmErrorHandler);
stream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
stream.client = new CustomClient();
video.attachNetStream(stream);
stream.play(videoURL);
```

DRMContentData und Sitzungsobjekte

Wenn `DRMContentData` erstellt wird, wird es als Sitzungsobjekt verwendet, das auf das Flash Player DRM-Modul verweist. Alle `DRMManager`-APIs, die diese `DRMContentData` erhalten, verwenden dieses bestimmte DRM-Modul. Es gibt jedoch 2 `DRMManager`-APIs, die `DRMContentData` nicht verwenden. Diese sind nachfolgend beschrieben:

1 `authenticate()`

2 `setAuthenticationToken()`

Da es kein zugeordnetes `DRMContentData`-Objekt gibt, führt der Aufruf dieser `DRMManager`-APIs dazu, dass das neueste DRM-Modul von der Festplatte verwendet wird. Dies kann problematisch sein, wenn mitten im DRM-Arbeitsablauf der Anwendung eine Aktualisierung des DRM-Moduls erfolgt. Betrachten Sie folgendes Szenario:

1 Die Anwendung erstellt ein `DRMContentData`-Objekt, `contentData1`, das *AdobeCP1* als DRM-Modul verwendet.

2 Die Anwendung ruft die `DRMManager.authenticate(contentData1.serverURL,...)`-Methode auf.

3 Die Anwendung ruft die `DRMManager.loadVoucher(contentData1, ...)`-Methode auf.

Wenn das DRM-Modul aktualisiert wird, bevor die Anwendung Schritt 2 erreicht, verwendet die `DRMManager.authenticate()`-Methode *AdobeCP2* als DRM-Modul für die Authentifizierung. Die `loadVoucher()`-Methode in Schritt 3 schlägt fehl, da sie immer noch *AdobeCP1* als DRM-Modul verwendet. Die Aktualisierung kann erfolgt sein, weil eine andere Anwendung die DRM-Modul-Aktualisierung aufgerufen hat. Sie können dies vermeiden, indem Sie die DRM-Modul-Aktualisierung beim Starten der Anwendung aufrufen.

DRM-Ereignisse

Die Laufzeit löst zahlreiche Ereignisse aus, wenn eine Anwendung versucht, geschützten Inhalt wiederzugeben:

- DRMDeviceGroupErrorEvent (nur AIR), abgesetzt von DRMManager
- DRMAuthenticateEvent (nur AIR), von NetStream ausgelöst
- DRMAuthenticationCompleteEvent, von DRMManager ausgelöst
- DRMAuthenticationErrorEvent, von DRMManager ausgelöst
- DRMErrorEvent, von NetStream und DRMManager ausgelöst
- DRMStatusEvent, von NetStream und DRMManager ausgelöst
- StatusEvent
- NetStatusEvent. Siehe „Warten auf ein Aktualisierungsereignis“ auf Seite 580.

Zur Unterstützung von Inhalten, die durch Adobe Access geschützt sind, fügen Sie Ereignis-Listener zur Verarbeitung der DRM-Ereignisse hinzu.

Vorausladen von Gutscheinen für die Offlinewiedergabe

Adobe AIR 1.5 und höher

Sie können die Gutscheine (Lizenzen), die zum Abspielen von durch Adobe Access geschützten Inhalten erforderlich sind, im Voraus laden. Mithilfe von im Voraus geladenen Gutscheinen können Benutzer Inhalte anzeigen, auch wenn sie keine aktive Internetverbindung haben. (Für das Vorausladen selbst ist jedoch eine Internetverbindung erforderlich.) Zum Vorausladen von Gutscheinen können Sie die `preloadEmbeddedMetadata()`-Methode der NetStream-Klasse und die DRMManager-Klasse verwenden. In AIR 2.0 und höher können Sie ein DRMContentData-Objekt verwenden, um Gutscheine direkt im Voraus zu laden. Diese Technik ist vorzuziehen, da das DRMContentData-Objekt unabhängig vom Inhalt aktualisiert werden kann. (Die `preloadEmbeddedData()`-Methode ruft DRMContentData aus dem Inhalt ab.)

Verwenden von DRMContentData

Adobe AIR 2.0 und höher

Im Folgenden wird der Arbeitsablauf zum Vorausladen des Gutscheins für eine geschützte Mediendatei mithilfe eines DRMContentData-Objekts beschrieben.

1. Rufen Sie die binären Metadaten für den verpackten Inhalt ab. Bei Verwendung des Adobe Access Java Reference Packager wird diese Metadaten-Datei automatisch mit der *.metadata*-Erweiterung erstellt. Sie können diese Metadaten beispielsweise mit der URLLoader-Klasse herunterladen.
2. Erstellen Sie ein DRMContentData-Objekt und übergeben Sie die Metadaten an die Konstruktorfunktion:

```
var drmData:DRMContentData = new DRMContentData( metadata );
```

3. Die restlichen Arbeitsschritte sind mit dem Arbeitsablauf identisch, der unter „Arbeitsablauf für geschützte Inhalte“ auf Seite 563 beschrieben wird.

Verwenden von preloadEmbeddedMetadata()

Adobe AIR 1.5 und höher

Die folgenden Schritte beschreiben den Arbeitsablauf beim Vorausladen des Gutscheins für eine DRM-geschützte Mediendatei mithilfe von `preloadEmbeddedMetadata()`:

1. Laden Sie die Mediendatei herunter und speichern Sie sie. (DRM-Metadaten können nur aus lokal gespeicherten Dateien vorausgeladen werden.)
2. Erstellen Sie die NetConnection- und NetStream-Objekte und stellen Sie Implementierungen für die `onDRMContentData()`- und `onPlayStatus()`-Rückruffunktionen des NetStream-Clientobjekts bereit.
3. Erstellen Sie ein NetStreamPlayOptions-Objekt und stellen Sie die `stream`-Eigenschaft auf die URL der lokalen Mediendatei ein.
4. Rufen Sie die NetStream-Methode `preloadEmbeddedMetadata()` auf und übergeben Sie dabei das NetStreamPlayOptions-Objekt, das die zu analysierende Mediendatei angibt.
5. Wenn die Mediendatei DRM-Metadaten enthält, wird die `onDRMContentData()`-Rückruffunktion aufgerufen. Die Metadaten werden als ein DRMContentData-Objekt an diese Funktion übergeben.
6. Verwenden Sie das DRMContentData-Objekt, um den Gutschein mit der `loadVoucher()`-Methode zu erhalten.

Wenn die `authenticationMethod`-Eigenschaft des `DRMContentData`-Objekts den Wert `flash.net.drm.AuthenticationMethod.USERNAME_AND_PASSWORD` hat, authentifizieren Sie den Benutzer auf dem Medienrechteserver, bevor der Gutschein geladen wird. Die Eigenschaften `serverURL` und `domain` des DRMContentData-Objekts können zusammen mit den Anmeldedaten des Benutzers an die `authenticate()`-Methode des DRMManager übergeben werden.

7 Die `onPlayStatus()`-Rückruffunktion wird aufgerufen, wenn das Dateiparsing abgeschlossen ist. Wenn die `onDRMContentData()`-Funktion nicht aufgerufen wurde, enthält die Datei nicht die für den Erhalt eines Gutscheins erforderlichen Metadaten. Dieser fehlende Aufruf kann auch bedeuten, dass diese Datei nicht von Adobe Access geschützt wird.

Im folgenden Beispielcode für AIR wird veranschaulicht, wie ein Gutschein für eine lokale Mediendatei vorausgeladen wird:

```
package
{
import flash.display.Sprite;
import flash.events.DRMAuthenticationCompleteEvent;
import flash.events.DRMAuthenticationErrorEvent;
import flash.events.DRMErrorEvent;
import flash.ev ents.DRMStatusEvent;
import flash.events.NetStatusEvent;
import flash.net.NetConnection;
import flash.net.NetStream;
import flash.net.NetStreamPlayOptions;
import flash.net.drm.AuthenticationMethod;
import flash.net.drm.DRMContentData;
import flash.net.drm.DRMManager;
import flash.net.drm.LoadVoucherSetting;
public class DRMPreloader extends Sprite
{
    private var videoURL:String = "app-storage:/video.flv";
    private var userName:String = "user";
    private var password:String = "password";
    private var preloadConnection:NetConnection;
    private var preloadStream:NetStream;
    private var drmManager:DRMManager = DRMManager.getDRMManager();
    private var drmContentData:DRMContentData;
    public function DRMPreloader():void {
        drmManager.addEventListener(
            DRMAuthenticationCompleteEvent.AUTHENTICATION_COMPLETE,
            onAuthenticationComplete);
        drmManager.addEventListener(DRMAuthenticationErrorEvent.AUTHENTICATION_ERROR,
            onAuthenticationError);
        drmManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
        drmManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
        preloadConnection = new NetConnection();
        preloadConnection.addEventListener(NetStatusEvent.NET_STATUS, onConnect);
        preloadConnection.connect(null);
    }

    private function onConnect( event:NetStatusEvent ):void
    {
        preloadMetadata();
    }
    private function preloadMetadata():void
    {
        preloadStream = new NetStream( preloadConnection );
```

```
        preloadStream.client = this;
        var options:NetStreamPlayOptions = new NetStreamPlayOptions();
        options.streamName = videoURL;
        preloadStream.preloadEmbeddedData( options );
    }
    public function onDRMContentData( drmMetadata:DRMContentData ):void
    {
        drmContentData = drmMetadata;
        if ( drmMetadata.authenticationMethod == AuthenticationMethod.USERNAME_AND_PASSWORD )
        {
            authenticateUser();
        }
        else
            {
                getVoucher();
            }
    }
    private function getVoucher():void
    {
        drmManager.loadVoucher( drmContentData, LoadVoucherSetting.ALLOW_SERVER );
    }

    private function authenticateUser():void
    {
        drmManager.authenticate( drmContentData.serverURL, drmContentData.domain, userName,
password );
    }
    private function onAuthenticationError( event:DRMAuthenticationErrorEvent ):void
    {
        trace( "Authentication error: " + event.errorID + ", " + event.subErrorID );
    }

    private function onAuthenticationComplete( event:DRMAuthenticationCompleteEvent ):void
    {
        trace( "Authenticated to: " + event.serverURL + ", domain: " + event.domain );
        getVoucher();
    }
    private function onDRMStatus( event:DRMStatusEvent ):void
    {
        trace( "DRM Status: " + event.detail);
        trace("--Voucher allows offline playback = " + event.isAvailableOffline );
        trace("--Voucher already cached          = " + event.isLocal );
        trace("--Voucher required authentication = " + !event.isAnonymous );
    }
    private function onDRMError( event:DRMErrorEvent ):void
    {
        trace( "DRM error event: " + event.errorID + ", " + event.subErrorID + ", " + event.text );
    }
    public function onPlayStatus( info:Object ):void
    {
        preloadStream.close();
    }
}
}
```

DRM-relevante Mitglieder und Ereignisse der NetStream-Klasse

Flash Player 10.1 und höher, Adobe AIR 1.0 und höher

Die NetStream-Klasse stellt eine unidirektionale Streaming-Verbindung zwischen Flash Player oder einer AIR-Anwendung und Flash Media Server oder dem lokalen Dateisystem zur Verfügung. (Die NetStream-Klasse unterstützt auch progressive Downloads.) Ein NetStream-Objekt ist ein Kanal innerhalb eines NetConnection-Objekts. Die NetStream-Klasse löst vier Ereignisse in Bezug auf DRM aus:

Ereignis	Beschreibung
drmAuthenticate (Nur AIR)	In der DRMAuthenticateEvent-Klasse definiert. Dieses Ereignis wird ausgelöst, wenn ein NetStream-Objekt versucht, geschützten Inhalt wiederzugeben, für den vor der Wiedergabe eine Benutzeranmeldung zur Authentifizierung erforderlich ist. Zu den Eigenschaften dieses Ereignisses zählen header, usernamePrompt, passwordPrompt und urlPrompt. Hiermit können die Benutzerinformationen eingeholt und eingestellt werden. Das Ereignis tritt so lange auf, bis das NetStream-Objekt gültige Benutzerinformationen erhält.
drmError	Ist in der DRMErrorEvent-Klasse definiert und wird ausgelöst, wenn ein NetStream-Objekt versucht, geschützten Inhalt wiederzugeben, dabei aber ein DRM-Fehler auftritt. Ein DRM-Fehler wird zum Beispiel ausgelöst, wenn die Benutzerautorisierung fehlschlägt. Dieser Fehler kann auftreten, wenn der Benutzer keine Berechtigung zum Anzeigen des Inhalts erworben hat. Eine weitere Fehlerursache liegt vor, wenn der Inhaltsanbieter die Anzeigeanwendung nicht unterstützt.
drmStatus	In der DRMStatusEvent-Klasse definiert. Dieses Ereignis wird ausgelöst, wenn die Wiedergabe des geschützten Inhalts beginnt (wenn der Benutzer authentifiziert wurde und zum Abspielen des Inhalts berechtigt ist). Das DRMStatusEvent-Objekt enthält Informationen zum Gutschein. Zu den Gutscheininformationen zählen Angaben dazu, ob die Inhalte offline bereitgestellt werden können oder wann der Gutschein abläuft und die Inhalte nicht mehr angezeigt werden können.
status	Ist in events.StatusEvent definiert und wird nur ausgelöst, wenn die Anwendung versucht, geschützten Inhalt durch Aufrufen der NetStream.play()-Methode wiederzugeben. Der Wert der Statuscode-Eigenschaft lautet „DRM.encryptedFLV“.

Die NetStream-Klasse enthält die folgenden DRM-spezifischen Methoden nur zur Verwendung in AIR:

Methode	Beschreibung
resetDRMVouchers()	Löscht alle lokal zwischengespeicherten DRM-Gutscheindaten (DRM = Digital Rights Management). Die Anwendung muss die Gutscheine erneut herunterladen, damit der Benutzer auf die verschlüsselten Inhalte zugreifen kann. Mit dem folgenden Code werden zum Beispiel alle Gutscheine aus dem Cache-Speicher entfernt: NetStream.resetDRMVouchers();
setDRMAuthenticationCredentials()	Übergibt die Authentifizierungsinformationen (also Benutzername, Kennwort und Authentifizierungstyp) zur Authentifizierung an das NetStream-Objekt. Folgende Authentifizierungstypen sind gültig: `„drm"` und `„proxy"` . Mit dem Authentifizierungstyp `„drm"` werden die Benutzerinformationen für Adobe Access authentifiziert. Beim Authentifizierungstyp `„proxy"` werden die Benutzerinformationen für den Proxyserver authentifiziert und müssen mit den vom Proxyserver angeforderten Benutzerinformationen übereinstimmen. Beispielsweise kann ein Unternehmen verlangen, dass die Anwendung über einen Proxyserver authentifiziert wird, bevor der Benutzer auf das Internet zugreifen kann. Die Proxy-Option ermöglicht diese Art der Authentifizierung. Sofern keine anonyme Authentifizierung verwendet wird, muss sich der Benutzer nach der Proxyauthentifizierung immer noch für Adobe Access authentifizieren, um den Gutschein zu erhalten und den Inhalt abzuspielen. Sie können `setDRMAuthenticationCredentials()` ein zweites Mal verwenden, mit der `drm`-Option, um die Authentifizierung für Adobe Access auszuführen.
preloadEmbeddedMetadata()	Sucht in einer lokalen Mediendatei nach eingebetteten Metadaten. Wenn DRM-relevante Metadaten gefunden werden, ruft AIR die `onDRMContentData()`-Rückruffunktion auf.

Zusätzlich ruft ein NetStream-Objekt in AIR die Rückruffunktionen `onDRMContentData()` und `onPlayStatus()` als Ergebnis eines Aufrufs der `preloadEmbeddedMetaData()`-Methode auf. Die `onDRMContentData()`-Funktion wird aufgerufen, wenn in einer Mediendatei DRM-Metadaten gefunden werden. Die `onPlayStatus()`-Funktion wird aufgerufen, wenn die Datei analysiert wurde. Die Funktionen `onDRMContentData()` und `onPlayStatus()` müssen für das `client`-Objekt definiert sein, das der NetStream-Instanz zugeordnet ist. Wenn Sie dasselbe NetStream-Objekt verwenden, um Gutscheine vorauszuladen und Inhalte abzuspielen, warten Sie vor der Wiedergabe auf den Aufruf von `onPlayStatus()`, der von `preloadEmbeddedMetaData()` generiert wird.

Im folgenden Code für AIR wurden der Benutzername („administrator"), das Kennwort („password") und der Authentifizierungstyp „drm" zur Authentifizierung des Benutzers festgelegt. Die setDRMAuthenticationCredentials()-Methode muss Benutzerinformationen übergeben, die denen entsprechen, die beim Inhaltsanbieter bekannt sind und akzeptiert werden. Dies sind die Benutzerinformationen, die die Anzeige des Inhalts ermöglichen. Der Code für das Abspielen des Videos und zur Überprüfung der Verbindung zum Videostream werden hier nicht angegeben.

```
var connection:NetConnection = new NetConnection();
connection.connect(null);

var videoStream:NetStream = new NetStream(connection);

videoStream.addEventListener(DRMAuthenticateEvent.DRM_AUTHENTICATE,
                             drmAuthenticateEventHandler)

private function drmAuthenticateEventHandler(event:DRMAuthenticateEvent):void
{
    videoStream.setDRMAuthenticationCredentials("administrator", "password", "drm");
}
```

Verwenden der DRMStatusEvent-Klasse

Flash Player 10.1, Adobe AIR 1.0 und höher

Ein NetStream-Objekt löst ein DRMStatusEvent-Objekt aus, wenn die Wiedergabe des durch Adobe Access geschützten Inhalts erfolgreich beginnt. („Erfolg“ bedeutet, dass die Lizenz überprüft wurde und dass der Benutzer authentifiziert wurde und zum Anzeigen des Inhalts berechtigt ist.) DRMStatusEvent wird auch für anonyme Benutzer ausgelöst, wenn der Zugriff für diese gestattet ist. Die Lizenz wird überprüft, um zu verifizieren, ob anonyme Benutzer, für die keine Authentifizierung erforderlich ist, Zugriff auf die Inhalte haben. Der Zugriff kann anonymen Benutzern aus verschiedenen Gründen verweigert werden. So haben anonyme Benutzer beispielsweise keinen Zugriff auf die Inhalte, wenn die Lizenz abgelaufen ist.

Das DRMStatusEvent-Objekt enthält Informationen zur Lizenz. Zu diesen Informationen zählen Angaben dazu, ob die Lizenz offline bereitgestellt werden kann oder wann der Gutschein abläuft und die Inhalte nicht mehr angezeigt werden können. Die Anwendung kann diese Daten verwenden, um den Richtlinienstatus und die Berechtigungen des Benutzers zu übermitteln.

DRMStatusEvent-Eigenschaften

Flash Player 10.1, Adobe AIR 1.0 und höher

Die DRMStatusEvent-Klasse umfasst die folgenden Eigenschaften. Einige Eigenschaften wurden in AIR-Versionen nach 1.0 eingeführt. Ausführliche Versionsinformationen finden Sie im *ActionScript 3.0 Referenzhandbuch*.

Für Eigenschaften, die in Flash Player 10.1 nicht unterstützt werden, bietet die DRMVoucher-Klasse ähnliche Eigenschaften für Flash Player.

Eigenschaft	Beschreibung
contentData	Ein DRMContentData-Objekt, das die im Inhalt eingebetteten DRM-Metadaten enthält.
detail (nur AIR)	Ein String, der den Kontext des Statusereignisses erläutert. In DRM 1.0 lautet der einzige gültige Wert DRM.voucherObtained.
isAnonymous (nur AIR)	Gibt an, ob die durch Adobe Access geschützten Inhalte ohne Eingabe von Authentifizierungsdaten verfügbar sind (true) oder nicht (false). Lautet der Wert „false“, muss der Benutzer Informationen (Benutzername und Kennwort) eingeben, die mit den Informationen übereinstimmen, die dem Anbieter bekannt sind und von diesem erwartet werden.
isAvailableOffline (nur AIR)	Gibt an, ob die durch Adobe Access geschützten Inhalte offline zur Verfügung gestellt werden können (true) oder nicht (false). Damit der digital geschützte Inhalt offline zur Verfügung steht, muss der entsprechende Gutschein auf dem lokalen Computer des Benutzer zwischengespeichert sein.
isLocal	Gibt an, ob der zum Abspielen des Inhalts erforderliche Gutschein lokal zwischengespeichert wird.
offlineLeasePeriod (nur AIR)	Die verbleibende Anzahl der Tage, an denen die Inhalte offline angezeigt werden können.
policies (nur AIR)	Ein benutzerdefiniertes Objekt, das benutzerdefinierte DRM-Eigenschaften enthalten kann.
voucher	Das DRMVoucher-Objekt (der DRM-Gutschein).
voucherEndDate (nur AIR)	Das absolute Datum, an dem der Gutschein abläuft und die Inhalte nicht mehr eingesehen werden können.

Erstellen von DRMStatusEvent-Prozeduren

Flash Player 10.1, Adobe AIR 1.0 und höher

Im folgenden Beispiel wird eine Ereignisprozedur erstellt, mit der Statusinformationen für DRM-Inhalte für das NetStream-Objekt ausgegeben werden, welches das Ereignis ausgelöst hat. Fügen Sie diese Ereignisprozedur einem NetStream-Objekt hinzu, das auf geschützte Inhalte verweist.

```
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
```

Verwenden der DRMAuthenticateEvent-Klasse

Adobe AIR 1.0 und höher

Das DRMAuthenticateEvent-Objekt wird ausgelöst, wenn ein NetStream-Objekt versucht, geschützten Inhalt wiederzugeben, für den vor der Wiedergabe eine Benutzeranmeldung zur Authentifizierung erforderlich ist.

Die DRMAuthenticateEvent-Prozedur ist zuständig für das Sammeln der erforderlichen Benutzerdaten (Benutzername, Kennwort und Typ) und die Übergabe der Werte an die `NetStream.setDRMAuthenticationCredentials()`-Methode zur Validierung. Jede AIR-Anwendung muss einen Mechanismus zur Verfügung stellen, mit dem die Benutzerinformationen erhalten werden. Beispielsweise kann eine Anwendung eine einfache Oberfläche enthalten, über die der Benutzer seinen Benutzernamen und sein Kennwort eingeben kann. Stellen Sie auch einen Mechanismus zum Verarbeiten und Beschränken wiederholter Authentifizierungsversuche bereit.

DRMAuthenticateEvent-Eigenschaften

Adobe AIR 1.0 und höher

Die Klasse DRMAuthenticateEvent umfasst die folgenden Eigenschaften:

Eigenschaft	Beschreibung
authenticationType	Gibt an, ob die angegebenen Benutzerinformationen für die Authentifizierung durch Adobe Access („drm“) oder einen Proxyserver („proxy“) bestimmt sind. Die proxy-Option ermöglicht der Anwendung zum Beispiel die Authentifizierung über einen Proxyserver, sofern erforderlich, bevor der Benutzer Zugriff auf das Internet hat. Sofern keine anonyme Authentifizierung verwendet wird, muss sich der Benutzer nach der Proxyauthentifizierung immer noch für Adobe Access authentifizieren, um den Gutschein zu erhalten und den Inhalt abzuspielen. Sie können setDRMAuthenticationcredentials() ein zweites Mal verwenden, mit der drm-Option, um die Authentifizierung für Adobe Access auszuführen.
header	Der verschlüsselte Inhaltsdateiheader, der vom Server bereitgestellt wird. Er enthält Informationen über den Kontext der verschlüsselten Inhalte. Dieser Headerstring kann an die Flash-Anwendung übergeben werden, damit die Anwendung ein Dialogfeld für Benutzernamen und Kennwort konstruieren kann. Der Headerstring kann für Dialogfeldanleitungen verwendet werden. Der Header könnte zum Beispiel lauten: „Bitte geben Sie Ihren Benutzernamen und Ihr Kennwort ein“.
netstream	Das NetStream-Objekt, das dieses Ereignis eingeleitet hat.
passwordPrompt	Eine vom Server bereitgestellte Eingabeaufforderung für das Kennwort. Der String kann Anweisungen enthalten, die den erforderlichen Typ des Kennworts betreffen.
urlPrompt	Eine vom Server bereitgestellte Eingabeaufforderung für eine URL. Der String kann angeben, wohin Benutzername und Kennwort gesendet werden.
usernamePrompt	Eine vom Server bereitgestellte Eingabeaufforderung für den Benutzernamen. Der String kann Anweisungen enthalten, die den erforderlichen Typ des Benutzernamens betreffen. Ein Content-Provider könnte zum Beispiel eine E-Mail-Adresse als Benutzernamen verlangen.

Die zuvor erwähnten Strings werden nur vom FMRMS-Server bereitgestellt. Adobe Access Server verwendet diese Strings nicht.

Erstellen von DRMAuthenticateEvent-Prozeduren

Adobe AIR 1.0 und höher

Im folgenden Beispiel wird eine Ereignisprozedur erstellt, mit der ein Satz hartkodierter Authentifizierungsinformationen an das NetStream-Objekt gesendet werden, welches das Ereignis auslöste. (Der Code für das Abspielen des Videos und zur Überprüfung der Verbindung zum Videostream werden hier nicht angegeben.)

```
var connection:NetConnection = new NetConnection();
connection.connect(null);

var videoStream:NetStream = new NetStream(connection);

videoStream.addEventListener(DRMAuthenticateEvent.DRM_AUTHENTICATE,
                             drmAuthenticateEventHandler)

private function drmAuthenticateEventHandler(event:DRMAuthenticateEvent):void
{
    videoStream.setDRMAuthenticationCredentials("administrator", "password", "drm");
}
```

Erstellen von Oberflächen für die Eingabe von Benutzerinformationen

Adobe AIR 1.0 und höher

Wenn für geschützte Inhalte eine Authentifizierung der Benutzerinformationen erforderlich ist, muss die AIR-Anwendung diese Informationen in der Regel über eine Benutzeroberfläche vom Benutzer einholen.

Im Folgenden wird ein Flex-Beispiel für eine einfache Benutzeroberfläche für das Einholen von Benutzerinformationen dargestellt. Es besteht aus einem Bedienfeldobjekt, das zwei TextInput-Objekte umfasst, eines für den Benutzernamen und eines für das Kennwort. Außerdem enthält das Bedienfeld eine Schaltfläche, mit der die Methode `credentials()` gestartet wird.

```
<mx:Panel x="236.5" y="113" width="325" height="204" layout="absolute" title="Login">
    <mx:TextInput x="110" y="46" id="uName"/>
    <mx:TextInput x="110" y="76" id="pWord" displayAsPassword="true"/>
    <mx:Text x="35" y="48" text="Username:"/>
    <mx:Text x="35" y="78" text="Password:"/>
    <mx:Button x="120" y="115" label="Login" click="credentials()"/>
</mx:Panel>
```

`credentials()` ist eine benutzerdefinierte Methode, die die Werte für den Benutzernamen und das Kennwort an die `setDRMAuthenticationCredentials()`-Methode übergibt. Sobald die Werte übergeben wurden, setzt die `credentials()`-Methode die Werte der TextInput-Objekte zurück.

```
<mx:Script>
    <![CDATA[
        public function credentials():void
        {
            videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
            uName.text = "";
            pWord.text = "";
        }
    ]]>
</mx:Script>
```

Ein Verfahren zur Implementierung einer solchen einfachen Benutzeroberfläche ist die Aufnahme des Bedienfelds als Teil eines neuen Zustands. Der neue Zustand löst den Grundzustand ab, wenn das DRMAuthenticateEvent-Objekt ausgelöst wird. Das folgende Beispiel enthält ein VideoDisplay-Objekt mit einem Quellattribut, das auf eine geschützte FLV-Datei verweist. In diesem Fall wird die `credentials()`-Methode so modifiziert, dass sie auch die Anwendung in den Grundzustand zurückversetzt, und zwar nach dem Übergeben der Benutzerinformationen und Zurücksetzen der Werte für das TextInput-Objekt.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute"
    width="800"
    height="500"
    title="DRM FLV Player"
    creationComplete="initApp()" >

    <mx:states>
        <mx:State name="LOGIN">
            <mx:AddChild position="lastChild">
                    <mx:Panel x="236.5" y="113" width="325" height="204" layout="absolute"
                            title="Login">
                    <mx:TextInput x="110" y="46" id="uName"/>
                    <mx:TextInput x="110" y="76" id="pWord" displayAsPassword="true"/>
                    <mx:Text x="35" y="48" text="Username:"/>
                    <mx:Text x="35" y="78" text="Password:"/>
                    <mx:Button x="120" y="115" label="Login" click="credentials()"/>
                    <mx:Button x="193" y="115" label="Reset" click="uName.text='';
                            pWord.text='';"/>
                </mx:Panel>
            </mx:AddChild>
        </mx:State>
    </mx:states>

    <mx:Script>
        <![CDATA[
                import flash.events.DRMAuthenticateEvent;
            private function initApp():void
            {
                videoStream.addEventListener(DRMAuthenticateEvent.DRM_AUTHENTICATE,
                                            drmAuthenticateEventHandler);
            }

            public function credentials():void
            {
                videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
                uName.text = "";
                pWord.text = "";
                currentState='';
            }

            private function drmAuthenticateEventHandler(event:DRMAuthenticateEvent):void
            {
                currentState='LOGIN';
            }
        ]]>
    </mx:Script>

    <mx:VideoDisplay id="video" x="50" y="25" width="700" height="350"
        autoPlay="true"
        bufferTime="10.0"
        source="http://www.example.com/flv/Video.flv" />
</mx:WindowedApplication>
```

Verwenden der DRMErrorEvent-Klasse

Flash Player 10.1 und höher, Adobe AIR 1.0 und höher

Adobe Flash Player und Adobe AIR lösen ein DRMErrorEvent-Objekt aus, wenn ein NetStream-Objekt beim Versuch, geschützten Inhalt wiederzugeben, einen DRM-Fehler erkennt. Wenn die Benutzerinformationen in einer AIR-Anwendung ungültig sind, wird das DRMAuthenticateEvent-Objekt wiederholt ausgelöst, bis der Benutzer gültige Informationen eingibt oder bis die Anwendung weitere Anmeldeversuche verweigert. Es ist Aufgabe der Anwendung, alle anderen DRM-Fehlerereignisse zu überwachen, um die DRM-bezogenen Fehler zu erkennen, zu identifizieren und zu verarbeiten.

Selbst bei gültigen Benutzerinformationen können die Bedingungen des Gutscheins verhindern, dass der Benutzer den verschlüsselten Inhalt anzeigen kann. Beispielsweise kann der Zugriff verweigert werden, wenn der Benutzer versucht, Inhalt in einer nicht autorisierten Anwendung anzuzeigen. Dies ist eine Anwendung, die vom Herausgeber der verschlüsselten Inhalte nicht validiert wurde. In diesem Fall wird ein DRMErrorEvent-Objekt ausgelöst.

Die Fehlerereignisse können auch ausgelöst werden, wenn die Inhalte beschädigt sind oder wenn die Version der Anwendung nicht mit der auf dem Gutschein angegebenen Version übereinstimmt. Die Anwendung muss einen entsprechenden Mechanismus für die Verarbeitung der Fehler bereitstellen.

DRMErrorEvent-Eigenschaften

Flash Player 10.1 und höher, Adobe AIR 1.0 und höher

Eine ausführliche Fehlerliste finden Sie im Abschnitt zu Laufzeit-Fehlercodes im ActionScript 3.0 Referenzhandbuch. Die DRM-Fehler beginnen bei Fehlercode 3300.

Erstellen von DRMErrorEvent-Prozeduren

Flash Player 10.1 und höher, Adobe AIR 1.0 und höher

Im folgenden Beispiel wird eine Ereignisprozedur für das NetStream-Objekt erstellt, welches das Ereignis ausgelöst hat. Es wird aufgerufen, wenn das NetStream-Objekt beim Versuch, geschützten Inhalt wiederzugeben, einen Fehler erkennt. Gewöhnlich führt eine Anwendung beim Auftreten eines Fehlers verschiedene Bereinigungsaufgaben durch. Dann teilt sie dem Benutzer den Fehler mit und bietet Optionen zur Fehlerbehebung an.

```
private function drmErrorEventHandler(event:DRMErrorEvent):void
{
    trace(event.toString());
}
```

Verwenden der DRMManager-Klasse

Flash Player 10.1 und höher, Adobe AIR 1.5 und höher

Mit der DRMManager-Klasse können Sie Gutscheine und Sitzungen auf Medienrechteservern in Anwendungen verwalten.

Gutscheinverwaltung (nur AIR)

Wenn ein Benutzer geschützten Inhalt abspielt, wird die dazu erforderliche Lizenz in der Laufzeit abgerufen und zwischengespeichert. Wenn die Anwendung die Datei lokal speichert und die Lizenz die Offlinewiedergabe zulässt, kann der Benutzer den Inhalt in der AIR-Anwendung anzeigen. Diese lokale Offlinewiedergabe ist auch dann möglich, wenn keine Verbindung mit dem Medienrechteserver verfügbar ist. Mit DRMManager und der `preloadEmbeddedMetadata()`-Methode von NetStream können Sie den Gutschein vorab zwischenspeichern. Die Anwendung muss die zur Anzeige des Inhalts erforderliche Lizenz nicht abrufen. Ihre Anwendung kann zum Beispiel die Mediendatei herunterladen und dann den Gutschein abrufen, während der Benutzer noch online ist.

Um einen Gutschein vorauszuladen, verwenden Sie die NetStream-Methode `preloadEmbeddedMetadata()`, um ein DRMContentData-Objekt zu erhalten. Das DRMContentData-Objekt enthält die URL und die Domäne des Medienrechteservers, der die Lizenz bereitstellen kann, und gibt an, ob die Benutzerauthentifizierung erforderlich ist. Mit diesen Informationen können Sie die DRMManager-Methode `loadVoucher()` aufrufen, um den Gutschein zu erhalten und zwischenzuspeichern. Der Ablauf beim Vorausladen von Gutscheinen wird ausführlicher unter „Vorausladen von Gutscheinen für die Offlinewiedergabe“ auf Seite 567 beschrieben.

Sitzungsverwaltung

Sie können den DRMManager auch verwenden, um den Benutzer für einen Medienrechteserver zu authentifizieren und dauerhafte Sitzungen zu verwalten.

Rufen Sie die DRMManager-Methode `authenticate()` auf, um eine Sitzung mit dem Medienrechteserver herzustellen. Wenn die Authentifizierung erfolgreich abgeschlossen wird, löst der DRMManager ein DRMAuthenticationCompleteEvent-Objekt aus. Dieses Objekt enthält ein Sitzungs-Token. Sie können dieses Token speichern, um es für zukünftige Sitzungen zu verwenden, damit der Benutzer seine Anmeldedaten nicht einzugeben braucht. Übergeben Sie das Token an die `setAuthenticationToken()`-Methode, um eine neue authentifizierte Sitzung herzustellen. (Die Gültigkeitsdauer des Tokens sowie andere Attribute werden von den Einstellungen des Servers, der das Token generiert hat, bestimmt. Die Datenstruktur des Tokens sollte nicht vom AIR-Anwendungscode interpretiert werden, da sie in späteren AIR-Aktualisierungen möglicherweise geändert wird.)

Authentifizierungs-Token können auf andere Computer übertragen werden. Um Token zu schützen, können Sie sie im verschlüsselten lokalen Speicher von AIR speichern. Weitere Informationen finden Sie unter „Verschlüsselter lokaler Speicher“ auf Seite 757.

DRMStatus-Ereignisse

Flash Player 10.1 und höher, Adobe AIR 1.5 und höher

Der DRMManager löst ein DRMStatusEvent-Objekt aus, nachdem die `loadVoucher()`-Methode erfolgreich aufgerufen wurde.

Wenn ein Gutschein abgerufen wird, hat die `detail`-Eigenschaft (nur AIR) des Ereignisobjekts den Wert „DRM.voucherObtained“ und die `voucher`-Eigenschaft enthält das DRMVoucher-Objekt.

Wenn kein Gutschein abgerufen wird, hat die `detail`-Eigenschaft (nur AIR) immer noch den Wert „DRM.voucherObtained“, doch die `voucher`-Eigenschaft ist `null`. Ein Gutschein kann nicht abgerufen werden, wenn Sie beispielsweise LoadVoucherSetting mit dem Wert *localOnly* verwenden und kein lokal gespeicherter Gutschein vorhanden ist.

Wenn der Aufruf von `loadVoucher()` nicht erfolgreich abgeschlossen wird, möglicherweise wegen eines Authentifizierungs- oder Kommunikationsfehlers, löst der DRMManager stattdessen ein DRMErrorEvent- oder DRMAuthenticationErrorEvent-Objekt aus.

DRMAuthenticationComplete-Ereignisse

Flash Player 10.1 und höher, Adobe AIR 1.5 und höher

Der DRMManager löst ein DRMAuthenticationCompleteEvent-Objekt aus, wenn ein Benutzer über einen Aufruf der `authenticate()`-Methode erfolgreich authentifiziert wird.

Das DRMAuthenticationCompleteEvent-Objekt enthält ein wiederverwendbares Token, mit dem die Benutzerauthentifizierung über mehrere Anwendungssitzungen erhalten bleibt. Übergeben Sie dieses Token an die `setAuthenticationToken()`-Methode des DRMManagers, um die Sitzung wiederherzustellen. (Die Tokenattribute, zum Beispiel die Gültigkeitsdauer, werden vom Tokenersteller festgelegt. Adobe bietet keine API zur Überprüfung der Tokenattribute.)

DRMAuthenticationError-Ereignisse

Flash Player 10.1 und höher, Adobe AIR 1.5 und höher

Der DRMManager löst ein DRMAuthenticationErrorEvent-Objekt aus, wenn ein Benutzer über einen Aufruf der Methode `authenticate()` oder `setAuthenticationToken()` nicht erfolgreich authentifiziert werden kann.

Verwenden der DRMContentData-Klasse

Flash Player 10.1 und höher, Adobe AIR 1.5 und höher

Das DRMContentData-Objekt enthält die Metadaten-Eigenschaften des durch Adobe Access geschützten Inhalts. Die DRMContentData-Eigenschaften enthalten die Informationen, die erforderlich sind, um einen Lizenzgutschein für das Anzeigen des Inhalts zu erhalten. Sie können die DRMContentData-Klasse verwenden, um die Metadaten-Datei des Inhalts abzurufen, wie unter „Ausführlicher API-Arbeitsablauf“ auf Seite 564 beschrieben.

Weitere Informationen finden Sie im Abschnitt zur DRMContentData-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Aktualisieren von Flash Player zur Unterstützung von Adobe Access

Flash Player 10.1 und höher

Wichtig: In Flash Player 11.5 und höheren Versionen ist das Adobe Access-Modul bereits enthalten, sodass der Updateschritt (Aufrufen von `SystemUpdater.update(SystemUpdaterType.DRM)`) nicht erforderlich ist. Dies schließt die folgenden Browser und Plattformen ein:

- Flash Player 11.5 ActiveX-Steuerelement, für alle Plattformen außer Internet Explorer unter Windows 8
- Flash Player 11.5-Plug-In, für alle Browser
- Adobe AIR (Desktop und Mobil)

Dies bedeutet, dass der Updateschritt in den folgenden Fällen *weiterhin erforderlich ist*:

- Internet Explorer unter Windows 8

- Flash Player 11.4 und älter, außer mit Google Chrome 22 und höher (alle Plattformen) oder 21 und höher (Windows)

Hinweis: *Es treten keine Probleme auf, wenn Sie* `SystemUpdater.update(SystemUpdaterType.DRM)` *in einem System mit Flash Player 11.5 oder höher aufrufen, es wird jedoch nichts heruntergeladen.*

Zur Unterstützung von Adobe Access benötigt Flash Player das Adobe Access-Modul. Wenn Flash Player versucht, geschützten Inhalt abzuspielen, gibt die Laufzeit an, ob das Modul oder eine neue Version von Flash Player heruntergeladen werden muss. So bietet Flash Player Entwicklern von SWF-Inhalten die Möglichkeit, auf eine Aktualisierung zu verzichten.

In den meisten Fällen aktualisieren SWF-Entwickler zum Abspielen von geschütztem Inhalt das erforderliche Adobe Access-Modul oder den benötigten Player. Zum Aktualisieren können Sie mit der SystemUpdater-API die neueste Version des Adobe Access-Moduls oder von Flash Player herunterladen.

Die SystemUpdater-API lässt jeweils nur eine Aktualisierung gleichzeitig zu. Fehlercode 2202 bedeutet, dass bereits eine Aktualisierung in der aktuellen Laufzeitinstanz oder in einer anderen Instanz durchgeführt wird. Wenn beispielsweise gerade eine Aktualisierung in einer Flash Player-Instanz in Internet Explorer durchgeführt wird, kann nicht gleichzeitig eine Aktualisierung in einer Flash Player-Instanz in Firefox vorgenommen werden.

Die SystemUpdater-API wird nur auf Desktop-Plattformen unterstützt.

Hinweis: *In Flash Player-Versionen vor 10.1 verwenden Sie den jeweils unterstützten Aktualisierungsmechanismus (manuelles Herunterladen und Installieren von www.adobe.com oder ExpressInstall). Beachten Sie auch, dass das AIR-Installationsprogramm die erforderlichen Aktualisierungen von Adobe Access durchführt und keine Unterstützung für die SystemUpdater-API bietet.*

Warten auf ein Aktualisierungsereignis

Flash Player 10.1 und höher

Wenn eine Aktualisierung des Adobe Access-Moduls erforderlich ist, löst das NetStream-Objekt ein NetStatusEvent-Ereignis mit dem Codewert `DRM.UpdateNeeded` aus. Dieser Wert weist darauf hin, dass das NetStream-Objekt den geschützten Stream nicht mit den derzeit installierten Adobe Access-Modulen wiedergeben kann. Warten Sie auf dieses Ereignis und rufen Sie den folgenden Code auf:

```
SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
```

Mit diesem Code wird das im Player installierte Adobe Access-Modul aktualisiert. Für diese Modulaktualisierung ist keine Zustimmung des Benutzers erforderlich.

Wenn das Adobe Access-Modul nicht gefunden wird, tritt ein Fehler auf. Siehe Schritt 3 unter „Ausführlicher API-Arbeitsablauf“ auf Seite 564.

Hinweis: *Wenn „play()“ in Player-Versionen vor 10.1 für einen verschlüsselten Stream aufgerufen wird, wird ein NetStatusEvent-Ereignis mit dem Codewert „NetStream.Play.StreamNotFound“ ausgelöst. In älteren Player-Versionen verwenden Sie den jeweils unterstützten Aktualisierungsmechanismus (manuelles Herunterladen und Installieren von www.adobe.com oder ExpressInstall).*

Wenn der Player selbst aktualisiert werden muss, löst das SystemUpdater-Objekt ein StatusEvent-Ereignis mit dem Codewert `DRM.UpdateNeededButIncompatible` aus. Für die Player-Aktualisierung ist die Zustimmung des Benutzers erforderlich. Stellen Sie in der Anwendung eine Oberfläche bereit, in der der Benutzer die Aktualisierung des Players akzeptieren und starten kann. Warten Sie auf das StatusEvent-Ereignis und rufen Sie den folgenden Code auf:

```
SystemUpdater.update(flash.system.SystemUpdaterType.SYSTEM);
```

Dieser Code leitet die Player-Aktualisierung ein.

Weitere Ereignisse der SystemUpdater-Klasse sind im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform dokumentiert.

Nachdem die Player-Aktualisierung abgeschlossen ist, kehrt der Benutzer zu der Seite zurück, auf der die Aktualisierung begann. Das Adobe Access-Modul wird heruntergeladen und der Stream kann nun wiedergegeben werden.

Out-of-Band-Lizenzen

Flash Player 11 und höher, Adobe AIR 3.0 und höher

Lizenzen können auch „out of band" (ohne Kontaktierung eines Adobe Access-Lizenzservers) bezogen werden, indem der Gutschein (die Lizenz) auf der Festplatte und im Arbeitsspeicher gespeichert wird. Hierzu wird die `storeVoucher`-Methode verwendet.

Um verschlüsselte Videos in Flash Player und AIR abzuspielen, muss die jeweilige Laufzeitumgebung den DRM-Gutschein für das Video beziehen. Der DRM-Gutschein enthält den Entschlüsselungsschlüssel des Videos und wird vom Adobe Access-Lizenzserver generiert, den der Kunde bereitgestellt hat.

Die Flash Player/AIR-Laufzeitumgebung bezieht diesen Gutschein im Allgemeinen durch das Senden einer Anforderung an den Adobe Access License Server, der in den DRM-Metadaten des Videos (`DRMContentData`-Klasse) angegeben ist. Die Flash-/AIR-Anwendung kann diese Lizenzanforderung auslösen, indem die `DRMManager.loadVoucher()`-Methode aufgerufen wird. Alternativ dazu fordert die Flash Player-/AIR-Laufzeitumgebung automatisch eine Lizenz an, wenn die Wiedergabe des verschlüsselten Videos gestartet wird, falls auf der Festplatte oder im Arbeitsspeicher keine Lizenz für den Inhalt vorhanden ist. In beiden Fällen wird die Leistung der Flash-/AIR-Anwendung durch die Kommunikation mit dem Adobe Access-Lizenzserver beeinträchtigt.

`DRMManager.storeVoucher()` ermöglicht der Flash-/AIR-Anwendung, DRM-Gutscheine an die Flash Player-/AIR-Laufzeitumgebung zu senden, die sie „out of band" bezogen hat. Die Laufzeitumgebung kann den Prozess der Lizenzanforderung überspringen und die weitergeleiteten Gutscheine für die Wiedergabe verschlüsselter Videos verwenden. Der DRM-Gutschein muss weiterhin vom Adobe Access-Lizenzserver generiert werden, bevor er „out of band" bezogen werden kann. Sie haben jedoch die Option, die Gutscheine auf einem beliebigen HTTP-Server bereitzustellen statt auf einem öffentlich ausgerichteten Adobe Access-Lizenzserver.

`DRMManager.storeVoucher()` wird auch verwendet, um die gemeinsame Nutzung von DRM-Gutscheinen auf mehreren Geräten zu unterstützen. In Adobe Access 3.0 wird diese Funktion als „Domänenunterstützung" bezeichnet. Wenn Ihre Implementierung diese Art der Verwendung unterstützt, können Sie mehrere Computer für eine Gerätegruppe registrieren, indem Sie die `DRMManager.addToDeviceGroup()`-Methode verwenden. Wenn es ein Gerät mit einem gültigen domänengebundenen Gutschein für einen bestimmten Inhalt gibt, kann die AIR-Anwendung die serialisierten DRM-Gutscheine mithilfe der `DRMVoucher.toByteArray()`-Methode extrahieren, und auf Ihren anderen Geräten können Sie die Gutscheine mithilfe der `DRMManager.storeVoucher()`-Methode importieren.

Geräteregistrierung

Die DRM-Gutscheine sind an das Gerät des Endbenutzers gebunden. Daher benötigen Flash-/AIR-Anwendungen eine eindeutige ID für das Gerät des Benutzers, um auf das richtige serialisierte DRM-Gutscheinobjekt zu verweisen. Das folgende Szenario stellt einen Geräteregistrierungsprozess dar:

Es wird davon ausgegangen, dass Sie die folgenden Aufgaben ausgeführt haben:

- Sie haben das Adobe Access Server SDK eingerichtet.

- Sie haben einen HTTP-Server für den Bezug der vorab generierten Lizenzen eingerichtet.
- Sie haben eine Flash-Anwendung erstellt, um die geschützten Inhalte anzuzeigen.

Die Phase der Geräteregistrierung umfasst die folgenden Aktionen:

1. Die Flash-Anwendung erstellt eine zufällig generierte ID.
2. Die Flash-Anwendung ruft die `DRMManager.authenticate()`-Methode auf. Die Anwendung muss die generierte Zufalls-ID in die Authentifizierungsanforderung einbeziehen. Die ID könnte zum Beispiel im Benutzernamenfeld erforderlich sein.
3. Die in Schritt 2 erwähnte Aktion führt dazu, dass Adobe Access eine Authentifizierungsanforderung an den Server des Kunden sendet. Diese Anforderung enthält das Gerätezertifikat.
   a. Der Server extrahiert das Gerätezertifikat und die generierte ID aus der Anforderung und speichert diese.
   b. Das Subsystem des Kunden generiert Lizenzen für dieses Gerätezertifikat im Voraus, speichert sie und gewährt auf eine Weise Zugriff darauf, die sie mit der generierten ID verknüpft.
4. Der Server antwortet mit einer Erfolgsmeldung auf die Anforderung.
5. Die Flash-Anwendung speichert die generierte ID lokal in einem Local Shared Object (LSO).

Nach der Geräteregistrierung verwendet die Flash-Anwendung die generierte ID auf die gleiche Weise wie die Geräte-ID im vorherigen Schema verwendet worden wäre:

1. Die Flash-Anwendung versucht, die generierte ID im LSO zu finden.
2. Wenn die generierte ID gefunden wird, verwendet die Flash-Anwendung diese ID, während die vorab generierten Lizenzen heruntergeladen werden. Die Flash-Anwendung sendet die Lizenzen an den Adobe Access-Client, der sie mit der `DRMManager.storeVoucher()`-Methode benutzt.
3. Wenn die generierte ID nicht gefunden wird, durchläuft die Flash-Anwendung das Verfahren der Geräteregistrierung.

Zurücksetzen auf Werkseinstellungen

Wenn der Benutzer des Geräts die Option zum Zurücksetzen auf die Werkseinstellungen verwendet, wird das Gerätezertifikat entfernt. Damit die geschützten Inhalte weiterhin abgespielt werden können, muss die Flash-Anwendung die Geräteregistrierung erneut ausführen. Wenn die Flash-Anwendung eine abgelaufene vorab generierte Lizenz sendet, wird diese vom Adobe Access-Client zurückgewiesen, da die Lizenz für eine ältere Geräte-ID verschlüsselt wurde.

Domänenunterstützung

Flash Player 11 und höher, Adobe AIR 3.0 und höher

Wenn die Metadaten des Inhalts angeben, dass die Domänenregistrierung erforderlich ist, kann die AIR-Anwendung eine API aufrufen, um einer Gerätegruppe beizutreten. Diese Aktion veranlasst, dass eine Domänenregistrierungsanforderung an den Domänenserver gesendet wird. Nachdem eine Lizenz an eine Gerätegruppe ausgegeben wurde, kann die Lizenz exportiert und mit anderen Geräten aus der Gerätegruppe gemeinsam genutzt werden.

Die Informationen zur Gerätegruppe werden dann im `VoucherAccessInfo`-Objekt von `DRMContentData` verwendet. Mit diesem Objekt werden dann die Informationen dargestellt, die zum erfolgreichen Abrufen und Benutzen eines Gutscheins erforderlich sind.

Abspielen verschlüsselter Inhalte mit Domänenunterstützung

Führen Sie die folgenden Schritte aus, um verschlüsselte Inhalte mit Adobe Access abzuspielen:

1. Überprüfen Sie mit `VoucherAccessInfo.deviceGroup` ob die Gerätegruppenregistrierung erforderlich ist.
2. Wenn Authentifizierung erforderlich ist:
   - **a** Stellen Sie anhand der `DeviceGroupInfo.authenticationMethod`-Eigenschaft fest, ob eine Authentifizierung erforderlich ist.
   - **b** Wenn eine Authentifizierung erforderlich ist, führen Sie dazu EINEN der folgenden Schritte aus:
     - Beziehen Sie den Benutzernamen und das Kennwort des Benutzers. Rufen Sie `DRMManager.authenticate(deviceGroup.serverURL, deviceGroup.domain, username, password)` auf.
     - Beziehen Sie ein im Cache gespeichertes/vorab generiertes Authentifizierungstoken und rufen Sie `DRMManager.setAuthenticationToken()` auf.
   - **c** Rufen Sie `DRMManager.addToDeviceGroup()` auf.
3. Rufen Sie den Gutschein für den Inhalt ab, indem Sie eine der folgenden Aufgaben ausführen:
   - **a** Verwenden Sie die `DRMManager.loadVoucher()`-Methode.
   - **b** Beziehen Sie den Gutschein von einem anderen Gerät, das in derselben Gerätegruppe registriert ist. Stellen Sie den Gutschein für `DRMManager` bereit, indem Sie die `DRMManager.storeVoucher()`-Methode verwenden.
4. Spielen Sie den verschlüsselten Inhalt mit der `NetStream.play()`-Methode ab.

Um die Lizenz für den Inhalt zu exportieren, kann jedes der Geräte mithilfe der `DRMVoucher.toByteArray()`-Methode die Raw-Bytes bereitstellen, nachdem die Lizenz vom Adobe Access-Lizenzserver bezogen wurde. Content-Provider legen normalerweise eine Höchstgrenze für die Anzahl der Geräte in einer Gerätegruppe fest. Wenn diese maximale Anzahl erreicht wird, müssen Sie ggf. die `DRMManager.removeFromDeviceGroup()`-Methode auf einem nicht verwendeten Gerät aufrufen, bevor Sie das aktuelle Gerät registrieren.

Lizenzvorschau

Die Flash-Anwendung kann eine Anforderung für eine Lizenzvorschau senden. Das bedeutet, dass die Anwendung eine Vorschauoperation ausführen kann, bevor der Benutzer aufgefordert wird, den Inhalt zu kaufen. So kann festgestellt werden, ob das Gerät des Benutzers alle Anforderungen für die Wiedergabe erfüllt. Die Lizenzvorschau bezieht sich auf die Fähigkeit des Clients, die Lizenz in einer Vorschau anzuzeigen (um festzustellen, welche Berechtigungen die Lizenz enthält), im Gegensatz zur Inhaltsvorschau (wobei der Benutzer vor dem Kauf einen kleinen Ausschnitt des Inhalts anzeigen kann). Zu den Parametern, die für jedes Gerät einzigartig sind, gehören folgende: verfügbare Ausgänge und ihr Schutzstatus, die verfügbare Laufzeitumgebung/DRM-Version und die Sicherheitsstufe des DRM-Clients. Im Lizenzvorschaumodus kann der Laufzeit-/DRM-Client die Geschäftslogik des Lizenzservers testen und dem Benutzer Informationen zur Verfügung stellen, damit dieser eine gut überlegte Entscheidung treffen kann. Auf diese Weise kann der Client prüfen, wie eine gültige Lizenz aussieht, erhält aber keinen Schlüssel zum Entschlüsseln des Inhalts. Die Unterstützung der Lizenzvorschau ist optional und nur dann erforderlich, wenn Sie eine benutzerdefinierte Anwendung implementieren, die diese Funktion verwendet.

Bereitstellen von Inhalten

Adobe Access beachtet die Zustellungsmechanismen der Inhalte nicht, da Flash Player die Netzwerkebene herausabstrahiert und dem Adobe Access-Subsystem lediglich die geschützten Inhalte bereitstellt. Somit kann der Inhalt über HTTP, HTTP Dynamic Streaming, RTMP oder RTMPE bereitgestellt werden.

Es kann jedoch aufgrund der Notwendigkeit der Metadaten des geschützten Inhalts (normalerweise in Form einer „.metadata"-Datei) Probleme geben, bevor Adobe Access eine Lizenz zum Entschlüsseln des Inhalts beziehen kann. Speziell mit dem RTMP/RTMPE-Protokoll können nur FLV- und F4V-Daten über den Flash Media Server (FMS) für den Client bereitgestellt werden. Deshalb muss der Client das Metadaten-Blob auf andere Weise abrufen. Eine Möglichkeit zur Lösung des Problems ist es, die Metadaten auf einem HTTP-Server zu hosten und den Client-Videoplayer zu implementieren, um die entsprechenden Metadaten abzurufen, je nach abgespieltem Inhalt.

```
private function getMetadata():void{

    extrapolated-path-to-metadata = "http://metadatas.mywebserver.com/" + videoname;
    var urlRequest : URLRequest = new URLRequest(extrapolated-path-to-the-metadata + ".metadata");
    var urlStream : URLStream = new URLStream();
    urlStream.addEventListener(Event.COMPLETE, handleMetadata);
    urlStream.addEventListener(IOErrorEvent.NETWORK_ERROR, handleIOError);
    urlStream.addEventListener(IOErrorEvent.IO_ERROR, handleIOError);
    urlStream.addEventListener(IOErrorEvent.VERIFY_ERROR, handleIOError);
    try{
        urlStream.load(urlRequest);
     }catch(se:SecurityError){
        videoLog.text += se.toString() + "\n";
     }catch(e:Error){
        videoLog.text += e.toString() + "\n";
     }
}
```

Open Source Media Framework

Open Source Media Framework (OSMF) ist ein ActionScript-basiertes Framework, das Ihnen vollständige Flexibilität und Kontrolle beim Erstellen eigener Rich-Media-Erlebnisse gibt. Weitere Informationen zu OSMF finden Sie auf der OSMF Developer Site.

Arbeitsablauf beim Abspielen geschützter Inhalte

1. Erstellen Sie eine `MediaPlayer`-Instanz.

   ```
   player = new MediaPlayer();
   ```

2. Registrieren Sie das `MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE`-Ereignis für den Player. Dieses Ereignis wird abgesetzt, wenn der Inhalt DRM-geschützt ist.

   ```
   player.addEventListener(MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE,
   onDRMCapabilityChange);
   ```

3. Rufen Sie in der Ereignisprozedur die `DRMTrait`-Instanz ab. `DRMTrait` ist die Schnittstelle, über die Sie DRM-bezogene Methoden wie zum Beispiel `authenticate()` aufrufen. Beim Laden DRM-geschützter Inhalten führt OSMF die DRM-Validierungsaktionen aus und setzt Statusereignisse ab. Fügen Sie dem `DRMTrait` eine `DRMEvent.DRM_STATE_CHANGE`-Ereignisprozedur hinzu.

```
private function onDRMCapabilityChange
    (event :MediaPlayerCapabilityChangeEvent) :void
        {
            if (event.type == MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE
                && event.enabled)
            {
                drmTrait = player.media.getTrait(MediaTraitType.DRM) as DRMTrait;
                drmTrait.addEventListener
                    (DRMEvent.DRM_STATE_CHANGE, onDRMStateChange);
            }
        }
```

4 Verarbeiten Sie die DRM-Ereignisse in der `onDRMStateChange()`-Methode.

```
private function onDRMStateChange(event :DRMEvent) :void
{
    trace ( "DRMState: ",event.drmState);
    switch(event.drmState)
    {
        case DRMState.AUTHENTICATION_NEEDED:
            // Identity-based content
            var authPopup :AuthWindow = AuthWindow.create(_parentWin);
            authPopup.serverURL = event.serverURL;
            authPopup.addEventListener("dismiss", function () :void {
                trace ("Authentication dismissed");
                if(_drmTrait != null)
                {
                    //Ignore authentication. Just
                    //try to acquire a license.
                    _drmTrait.authenticate(null, null);
                }
            });
            authPopup.addEventListener("authenticate",
                            function (event :AuthWindowEvent) :void {
                if(_drmTrait != null)
                {
                    _drmTrait.authenticate(event.username, event.password);
                }
            });
            authPopup.show();
            break;
        case DRMState.AUTHENTICATING:
            //Display any authentication message.
            trace("Authenticating...");
            break;
        case DRMState.AUTHENTICATION_COMPLETE:
            // Start to retrieve voucher and playback.
            // You can display the voucher information at this point.
            if(event.token)
            // You just received the authentication token.
            {
                trace("Authentication success. Token: \n", event.token);
            }
            else
            // You have got the voucher.
            {
                trace("DRM License:");
```

```
                    trace("Playback window period: ",
                        !isNaN(event.period) ? event.period == 0 ?
                        "<unlimited>" : event.period : "<none>");
                    trace("Playback window end date: ",
                        event.endDate != null ? event.endDate : "<none>");
                    trace("Playback window start date: ",
                        event.startDate != null ? event.startDate : "<none>");
                }
                break;
            case DRMState.AUTHENTICATION_ERROR:
                trace ("DRM Error:", event.mediaError.errorID +
                    "[" + DRMErrorEventRef.getDRMErrorMnemonic
                    (event.mediaError.errorID) + "]");
                //Stop everything.
                player.media = null;
                break;
            case DRMState.DRM_SYSTEM_UPDATING:
                Logger.log("Downloading DRM module...");
                break;
            case DRMState.UNINITIALIZED:
                break;
        }
    }
```

Kapitel 28: Hinzufügen von PDF-Inhalten in AIR

Adobe AIR 1.0 und höher

In Adobe® AIR® ausgeführte Dateien rendern nicht nur SWF- und HTML-Inhalt, sondern auch PDF-Inhalt. AIR-Anwendungen rendern PDF-Inhalt mit der HTMLLoader-Klasse, der WebKit-Engine und dem Browser-Plug-In für Adobe® Reader®. In einer AIR-Anwendung kann PDF-Inhalt die volle Höhe und Breite der Anwendung oder einen Teil der Benutzeroberfläche einnehmen. Das Browser-Plugin für Adobe Reader steuert die Anzeige von PDF-Dateien in einer AIR-Anwendung. Änderungen an der Reader-Symbolleiste (wie Positionssteuerung, Verankerung, Sichtbarkeit) werden bei der weiteren Anzeige von PDF-Dateien in AIR-Anwendungen und im Browser übernommen.

Wichtig: *Damit PDF-Inhalt in AIR gerendert werden kann, muss Adobe Reader oder Adobe® Acrobat® Version 8.1 oder höher auf dem Benutzersystem installiert sein.*

Erkennen der PDF-Funktionalität

Adobe AIR 1.0 und höher

Wenn der Benutzer Adobe Reader oder Adobe Acrobat 8.1 oder höher nicht installiert hat, wird in einer AIR-Anwendung kein PDF-Inhalt angezeigt. Um herauszufinden, ob ein Benutzer PDF-Inhalt rendern kann, prüfen Sie zunächst die `HTMLLoader.pdfCapability`-Eigenschaft. Für diese Eigenschaft ist eine der folgenden Konstanten der HTMLPDFCapability-Klasse festgelegt:

Konstante	Beschreibung
HTMLPDFCapability.STATUS_OK	Eine ausreichende Version (8.1 oder höher) von Adobe Reader wurde erkannt. PDF-Inhalt kann in ein HTMLLoader-Objekt geladen werden.
HTMLPDFCapability.ERROR_INSTALLED_READER_NOT_FOUND	Es wurde keine Version von Adobe Reader erkannt. Ein HTMLLoader-Objekt kann PDF-Inhalt nicht anzeigen.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	Adobe Reader wurde erkannt, die Version ist jedoch zu alt. Ein HTMLLoader-Objekt kann PDF-Inhalt nicht anzeigen.
HTMLPDFCapability.ERROR_PREFERRED_READER_TOO_OLD	Eine ausreichende Version (8.1 oder höher) von Adobe Reader wurde erkannt; die Version von Adobe Reader, die zum Verarbeiten von PDF-Inhalt eingerichtet wurde, ist jedoch älter als Reader 8.1. Ein HTMLLoader-Objekt kann PDF-Inhalt nicht anzeigen.

Wenn unter Windows Adobe Acrobat oder Adobe Reader Version 7.x oder höher installiert ist und auf dem System des Benutzers ausgeführt wird, wird diese Version verwendet, auch wenn eine höhere Version installiert ist, die das Laden von PDF-Dateien unterstützt. Hat die `pdfCapability`-Eigenschaft den Wert `HTMLPDFCapability.STATUS_OK`, zeigt in diesem Fall die alte Version von Acrobat oder Reader einen Warnhinweis an, wenn eine AIR-Anwendung versucht, PDF-Inhalt zu laden (in der AIR-Anwendung wird keine Ausnahme ausgelöst). Wenn diese Situation für die Endbenutzer auftreten kann, sollten Sie Anweisungen zum Schließen von Acrobat in Erwägung ziehen, die beim Ausführen der Anwendung angezeigt werden. Die Anzeige dieser Anweisungen sollte möglicherweise dann erfolgen, wenn der PDF-Inhalt nicht innerhalb eines akzeptablen Zeitraums geladen wird.

Unter Linux sucht AIR im vom Benutzer exportierten PATH nach Adobe Reader (falls dieser den acroread-Befehl enthält) und im Verzeichnis „/opt/Adobe/Reader“.

Mit dem folgenden Code wird festgestellt, ob der Benutzer PDF-Inhalt in einer AIR-Anwendung anzeigen kann. Wenn der Benutzer keinen PDF-Inhalt anzeigen kann, erfasst der Code den Fehlercode, der dem HTMLPDFCapability-Fehlerobjekt entspricht:

```
 if(HTMLLoader.pdfCapability == HTMLPDFCapability.STATUS_OK)
{
    trace("PDF content can be displayed");
}
else
{
    trace("PDF cannot be displayed. Error code:", HTMLLoader.pdfCapability);
}
```

Laden von PDF-Inhalten

Adobe AIR 1.0 und höher

Sie können einer AIR-Anwendung durch Erstellen einer HTMLLoader-Instanz, Festlegen ihrer Dimensionen und Laden des Pfades einer PDF-Datei PDF-Inhalt hinzufügen.

Das folgende Beispiel lädt eine PDF-Datei aus einer externen Site. Ersetzen Sie das URLRequest durch den Pfad zu einer verfügbaren externen PDF.

```
 var request:URLRequest = new URLRequest("http://www.example.com/test.pdf");
pdf = new HTMLLoader();
pdf.height = 800;
pdf.width = 600;
pdf.load(request);
container.addChild(pdf);
```

Sie können Inhalt ferner aus Datei-URLs und AIR-spezifischen URL-Schemas wie app und app-storage laden. Der folgende Code lädt die Datei „test.pdf“ im Unterverzeichnis „PDFs“ des Anwendungsverzeichnisses:

app:/js_api_reference.pdf

Weitere Informationen zu AIR-URL-Schemas finden Sie unter „URI-Schemas“ auf Seite 867.

Skripterstellung für PDF-Inhalte

Adobe AIR 1.0 und höher

Mit JavaScript können Sie PDF-Inhalt genau wie in einer Webseite in einem Browser steuern.

JavaScript-Erweiterungen für Acrobat bieten unter anderem folgende Funktionen:

- Steuern der Seitennavigation und -vergrößerung
- Verarbeiten von Formularen im Dokument
- Steuern von Multimediaereignissen

Ausführliche Informationen zu JavaScript-Erweiterungen für Adobe Acrobat finden Sie in der Adobe Acrobat Developer Connection unter http://www.adobe.com/devnet/acrobat/javascript.html.

Grundlagen der HTML-PDF-Kommunikation

Adobe AIR 1.0 und höher

JavaScript in einer HTML-Seite kann durch Aufrufen der `postMessage()`-Methode des DOM-Objekts, das den PDF-Inhalt repräsentiert, eine Nachricht an JavaScript in PDF-Inhalt senden. Betrachten Sie beispielsweise den folgenden eingebetteten PDF-Inhalt:

```
<object id="PDFObj" data="test.pdf" type="application/pdf" width="100%" height="100%"/>
```

Der folgende JavaScript-Code im enthaltenden HTML-Inhalt sendet eine Nachricht an das JavaScript in der PDF-Datei:

```
pdfObject = document.getElementById("PDFObj");
pdfObject.postMessage(["testMsg", "hello"]);
```

Die PDF-Datei kann JavaScript zum Empfangen dieser Nachricht enthalten. In einigen Kontexten, darunter Kontext auf Dokument-, Ordner-, Seiten-, Feld- oder Batchebene, können Sie PDF-Dateien JavaScript-Code hinzufügen. Hier wird nur auf Kontext auf Dokumentebene, der beim Öffnen des PDF-Dokuments zu bewertende Skripts definiert, eingegangen.

Eine PDF-Datei kann dem `hostContainer`-Objekt eine `messageHandler`-Eigenschaft hinzufügen. Die `messageHandler`-Eigenschaft ist ein Objekt, das Prozedurfunktionen zum Reagieren auf Nachrichten definiert. Der folgende Code definiert beispielsweise die Funktion zum Verarbeiten von Nachrichten, die die PDF-Datei vom Hostcontainer (dem HTML-Inhalt, der die PDF-Datei einbettet) erhält:

```
this.hostContainer.messageHandler = {onMessage: myOnMessage};

function myOnMessage(aMessage)
{
    if(aMessage[0] == "testMsg")
    {
        app.alert("Test message: " + aMessage[1]);
    }
    else
    {
        app.alert("Error");
    }
}
```

JavaScript-Code in der HTML-Seite kann die `postMessage()`-Methode des in der Seite enthaltenen PDF-Objekts aufrufen. Beim Aufrufen dieser Methode wird eine Nachricht (`"Hello from HTML"`) an das JavaScript auf Dokumentebene in der PDF-Datei gesendet:

```
<html>
    <head>
    <title>PDF Test</title>
    <script>
        function init()
        {
            pdfObject = document.getElementById("PDFObj");
            try {
                 pdfObject.postMessage(["alert", "Hello from HTML"]);
            }
            catch (e)
            {
                alert( "Error: \n name = " + e.name + "\n message = " + e.message );
            }
        }
    </script>
    </head>
    <body onload='init()'>
        <object
            id="PDFObj"
            data="test.pdf"
            type="application/pdf"
            width="100%" height="100%"/>
    </body>
</html>
```

Ein komplexeres Beispiel und Informationen zum Verwenden von Acrobat 8 zum Hinzufügen von JavaScript zu einer PDF-Datei finden Sie unter Cross-Scripting von PDF-Inhalt in Adobe AIR.

Skripterstellung für PDF-Inhalte aus ActionScript

Adobe AIR 1.0 und höher

ActionScript-Code (in SWF-Inhalt) kann nicht direkt mit JavaScript in PDF-Inhalt kommunizieren. ActionScript kann jedoch mit dem JavaScript in der in ein HTMLLoader-Objekt geladenen HTML-Seite, das PDF-Inhalt lädt, kommunizieren und dieser JavaScript-Code kann mit dem JavaScript in der geladenen PDF-Datei kommunizieren. Weitere Informationen finden Sie unter „Programmieren mit HTML und JavaScript in AIR“ auf Seite 1047.

Bekannte Beschränkungen für PDF-Inhalt in AIR

Adobe AIR 1.0 und höher

PDF-Inhalt in Adobe AIR unterliegt folgenden Beschränkungen:

- PDF-Inhalt wird nicht in einem transparenten Fenster (einem NativeWindow-Objekt) angezeigt, für dessen `transparent`-Eigenschaft `true` festgelegt wurde.

- Die Anzeigereihenfolge einer PDF-Datei unterscheidet sich von anderen Anzeigeobjekten in einer AIR-Anwendung. Zwar wird PDF-Inhalt entsprechend der HTML-Anzeigereihenfolge richtig dargestellt, doch befindet er sich in der Anzeigereihenfolge für Inhalt der AIR-Anwendung immer im Vordergrund.
- Wenn bestimmte visuelle Eigenschaften eines HTMLLoader-Objekts, das ein PDF-Dokument enthält, geändert werden, wird das PDF-Dokument unsichtbar. Zu diesen Eigenschaften gehören `filters`, `alpha`, `rotation` und `scaling`. Beim Ändern dieser Eigenschaften wird der PDF-Inhalt unsichtbar, bis die Eigenschaften zurückgesetzt werden. Der PDF-Inhalt ist auch unsichtbar, wenn Sie diese Eigenschaften für Anzeigeobjektcontainer ändern, die das HTMLLoader-Objekt enthalten.
- PDF-Inhalt ist nur dann sichtbar, wenn die `scaleMode`-Eigenschaft des Stage-Objekts des NativeWindow-Objekts mit dem PDF-Inhalt auf `StageScaleMode.NO_SCALE` eingestellt ist. Bei jedem anderen Wert ist der PDF-Inhalt nicht sichtbar.
- Durch Klicken auf Verknüpfungen zu Inhalten in der PDF-Datei wird die Bildlaufposition des PDF-Inhalts aktualisiert. Durch Klicken auf Verknüpfungen außerhalb der PDF-Datei wird das die PDF enthaltende HTMLLoader-Objekt umgeleitet (auch wenn das Ziel einer Verknüpfung ein neues Fenster ist).
- PDF-Kommentarworkflows können in AIR nicht eingesetzt werden.

Kapitel 29: Grundlagen der Benutzerinteraktion

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ihre Anwendung kann den Benutzer einbeziehen, indem mit ActionScript 3.0 auf Benutzeraktionen reagiert wird. Beachten Sie, dass in diesem Abschnitt vorausgesetzt wird, dass Sie mit dem Ereignismodell von ActionScript 3.0 vertraut sind. Weitere Informationen finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Erfassen von Benutzereingaben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Benutzerinteraktion per Tastatur, Maus, Kamera oder einer Kombination dieser Geräte ist die Grundlage von Interaktivität. In ActionScript 3.0 wird das Erkennen von und Reagieren auf Benutzerinteraktionen in erster Linie durch Warten auf Ereignisse realisiert.

Die InteractiveObject-Klasse ist eine Unterklasse der DisplayObject-Klasse und stellt die allgemeine Ereignisstruktur und Funktionen bereit, die zum Verarbeiten von Benutzerinteraktionen erforderlich sind. Es ist nicht erforderlich, dass Sie direkt eine Instanz der InteractiveObject-Klasse erstellen. Stattdessen erben Anzeigeobjekte wie SimpleButton, Sprite, TextField und verschiedene Flash-Authoring-Tool- und Flex-Komponenten das entsprechende Benutzerinteraktionsmodell von dieser Klasse und weisen deshalb eine gemeinsame Struktur auf. Das bedeutet, dass die erlernten Techniken und der für von der InteractiveObject-Klasse abgeleitete Objekte programmierte Code zum Verarbeiten von Benutzerinteraktionen auch bei allen anderen Objekten anwendbar sind.

Wichtige Konzepte und Begriffe

Es ist wichtig, dass Sie sich mit den folgenden Kernbegriffen zur Benutzerinteraktion vertraut machen, bevor Sie fortfahren:

Zeichencode Ein numerischer Code, der ein Zeichen aus dem aktuellen Zeichensatz darstellt (das einer auf der Tastatur gedrückten Taste zugeordnet ist). „D“ und „d“ haben beispielsweise unterschiedliche Zeichencodes, obwohl sie auf einer deutschen Tastatur durch Drücken derselben Taste erzeugt werden.

Kontextmenü Das Menü, das angezeigt wird, wenn ein Benutzer mit der rechten Maustaste klickt oder eine bestimmte Tastatur-Maus-Kombination verwendet. Kontextmenübefehle gelten üblicherweise direkt für das Element, auf das geklickt wurde. Beispielsweise kann ein Kontextmenü für ein Bild den Befehl zum Anzeigen des Bilds in einem separaten Fenster sowie einen Befehl zum Herunterladen des Bilds enthalten.

Fokus Eine visuelle Kennzeichnung, dass ein ausgewähltes Element aktiv und Ziel von Tastatur- oder Mausinteraktionen ist.

Tastencode Ein numerischer Code, der einer Taste auf der Tastatur entspricht.

Verwandte Hilfethemen

InteractiveObject

Keyboard

Mouse

ContextMenu

Fokusverwaltung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein interaktives Objekt kann den Fokus entweder durch Programmcode oder durch eine Benutzeraktion erhalten. Wenn die `tabEnabled`-Eigenschaft auf `true` gesetzt wird, können Benutzer zudem durch Drücken der TAB-Taste den Fokus zwischen Objekten weitergeben. Beachten Sie, dass der Wert für `tabEnabled` standardmäßig `false` ist, außer in den folgenden Fällen:

- Bei einem SimpleButton-Objekt lautet der Wert `true`.
- Bei Eingabetextfeldern ist dieser Wert `true`.
- Bei Sprite- oder MovieClip-Objekten, deren `buttonMode`-Eigenschaft auf `true` gesetzt ist, ist dieser Wert `true`.

In jedem dieser Fälle können Sie einen Listener für `FocusEvent.FOCUS_IN` oder `FocusEvent.FOCUS_OUT` hinzufügen, um zusätzliches Verhalten bei Fokusänderungen bereitzustellen. Dies ist besonders nützlich bei Textfeldern und Formularen, kann jedoch auch für Sprites, Movieclips oder andere Objekte eingesetzt werden, die von der InteractiveObject-Klasse erben. Im folgenden Beispiel ist dargestellt, wie Sie das zyklische Wechseln des Fokus über die TAB-Taste aktivieren und auf die daraufhin ausgelösten Fokusereignisse reagieren können. Im Beispiel ändert jedes der Rechtecke die Farbe, wenn es den Fokus erhält.

Hinweis: *In Flash Professional werden Tastaturbefehle für die Fokusverwaltung verwendet. SWF-Dateien sollten deshalb nicht in Flash, sondern in einem Browser oder in AIR getestet werden, um die Fokusverwaltung korrekt zu simulieren.*

```
var rows:uint = 10;
var cols:uint = 10;
var rowSpacing:uint = 25;
var colSpacing:uint = 25;
var i:uint;
var j:uint;
for (i = 0; i < rows; i++)
{
    for (j = 0; j < cols; j++)
    {
        createSquare(j * colSpacing, i * rowSpacing, (i * cols) + j);
    }
}

function createSquare(startX:Number, startY:Number, tabNumber:uint):void
{
    var square:Sprite = new Sprite();
    square.graphics.beginFill(0x000000);
    square.graphics.drawRect(0, 0, colSpacing, rowSpacing);
    square.graphics.endFill();
    square.x = startX;
```

```
    square.y = startY;
    square.tabEnabled = true;
    square.tabIndex = tabNumber;
    square.addEventListener(FocusEvent.FOCUS_IN, changeColor);
    addChild(square);
}
function changeColor(event:FocusEvent):void
{
    event.target.transform.colorTransform = getRandomColor();
}
function getRandomColor():ColorTransform
{
    // Generate random values for the red, green, and blue color channels.
    var red:Number = (Math.random() * 512) - 255;
    var green:Number = (Math.random() * 512) - 255;
    var blue:Number = (Math.random() * 512) - 255;

    // Create and return a ColorTransform object with the random colors.
    return new ColorTransform(1, 1, 1, 1, red, green, blue, 0);
}
```

Erkennen von Eingabetypen

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Ab Flash Player 10.1 und Adobe AIR 2 besteht die Möglichkeit, die Laufzeitumgebung hinsichtlich der Unterstützung bestimmter Eingabetypen zu testen. Mit ActionScript können Sie testen, ob das Gerät, auf dem die Laufzeit derzeit implementiert ist, folgende Kriterien erfüllt:

- Unterstützung für die Eingabe über einen Stift oder Finger (oder keine Berührungseingabe)
- Virtuelle oder physische Tastatur für den Benutzer (oder keine Tastatur)
- Anzeige eines Cursors (falls dies nicht der Fall ist, können Funktionsmerkmale, bei denen mit der Maus auf ein Objekt gezeigt werden muss, nicht verwendet werden)

Zu den ActionScript-APIs für die Eingabeerkennung gehören:

- flash.system.Capabilities.touchscreenType-Eigenschaft: Ein zur Laufzeit angegebener Wert, der anzeigt, welcher Eingabetyp in der aktuellen Umgebung unterstützt wird.
- flash.system.TouchscreenType-Klasse: Eine Klasse mit Aufzählungswertkonstanten für die Capabilities.touchScreenType-Eigenschaft.
- flash.ui.Mouse.supportsCursor-Eigenschaft: Ein zur Laufzeit angegebener Wert, der anzeigt, ob ein persistenter Cursor verfügbar ist oder nicht.
- flash.ui.Keyboard.physicalKeyboardType-Eigenschaft: Ein zur Laufzeit angegebener Wert, der anzeigt, ob eine vollständige physische Tastatur, nur ein numerisches Tastenfeld oder gar keine Tastatur verfügbar ist.
- flash.ui.KeyboardType-Klasse: Eine Klasse mit Aufzählungswertkonstanten für die flash.ui.Keyboard.physicalKeyboardType-Eigenschaft.
- flash.ui.Keyboard.hasVirtualKeyboard-Eigenschaft: Ein zur Laufzeit angegebener Wert, der anzeigt, ob dem Benutzer eine virtuelle Tastatur zur Verfügung gestellt wird (entweder anstatt oder zusätzlich zu einer physischen Tastatur).

Über die APIs für die Eingabeerkennung können Sie die Fähigkeiten des Endgeräts nutzen oder Alternativen bereitstellen, wenn keine solchen Fähigkeiten vorhanden sind. Diese APIs sind besonders bei der Entwicklung von Anwendungen für mobile und berührungsempfindliche Geräte hilfreich. Wenn die Benutzeroberfläche für ein Mobilgerät beispielsweise kleine Schaltflächen für einen Stift aufweist, können Sie eine alternative Benutzeroberfläche mit größeren Schaltflächen bereitstellen, die für die Eingabe per Finger geeignet ist. Der folgende Code ist für eine Anwendung mit einer Funktion namens „createStylusUI()“ vorgesehen. Diese Funktion weist Benutzeroberflächenelemente für eine stiftgesteuerte Interaktion zu. Eine weitere Funktion namens „createTouchUI()“ weist andere Benutzeroberflächenelemente zu, die für die Interaktion per Finger vorgesehen sind:

```
if(Capabilities.touchscreenType == TouchscreenType.STYLUS ){
    //Construct the user interface using small buttons for a stylus
    //and allow more screen space for other visual content
    createStylusUI();
} else if(Capabilities.touchscreenType = TouchscreenType.FINGER){
    //Construct the user interface using larger buttons
    //to capture a larger point of contact with the device
    createTouchUI();
}
```

Bei der Entwicklung von Anwendungen mit unterschiedlichen Eingabemechanismen sollten Sie die folgende Kompatibilitätsübersicht zurate ziehen:

Umgebung	supportsCursor	touchscreenType == FINGER	touchscreenType == STYLUS	touchscreenType == NONE
Herkömmlicher Desktop	true	false	false	true
Kapazitive Touchscreen-Geräte (Tabletts, PDAs und Telefone, die leichte Berührungen der Anwender erkennen, wie das Apple iPhone oder Palm Pre)	false	true	false	false
Resistive Touchscreen-Geräte (Tabletts, PDAs und Telefone, die präzisen Kontakt mit hohem Druck erkennen, wie das HTC Fuze)	false	false	true	false
Geräte ohne Touchscreen (Funktionstelefone und Geräte, auf denen Anwendungen ausgeführt werden, deren Bildschirme aber keinen Kontakt erkennen können)	false	false	false	true

Hinweis: *Verschiedene Geräteplattformen können zahlreiche Kombinationen aus Eingabetypen unterstützen. Dieses Diagramm soll als allgemeine Übersicht dienen.*

Kapitel 30: Tastatureingabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ihre Anwendung kann Tastatureingaben erfassen und darauf reagieren sowie ein IME bearbeiten, damit Benutzer Nicht-ASCII-Zeichen in Multibyte-Sprechen eingeben können. Beachten Sie, dass in diesem Abschnitt vorausgesetzt wird, dass Sie mit dem Ereignismodell von ActionScript 3.0 vertraut sind. Weitere Informationen finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Einzelheiten zur Erkennung der verfügbaren Tastaturunterstützung (physische, virtuelle, alphanumerische Tastatur oder numerische 12-Tasten-Tastatur) während der Laufzeit finden Sie unter „Erkennen von Eingabetypen“ auf Seite 594.

Mit einem Eingabemethodeneditor (Input Method Editor, IME) können Benutzer komplexe Zeichen und Symbole über eine Standardtastatur eingeben. Über die IME-Klassen geben Sie den Benutzern die Möglichkeit, ihren System-IME in Ihren Anwendungen zu nutzen.

Verwandte Hilfethemen

flash.events.KeyboardEvent

flash.system.IME

Erfassen von Tastatureingaben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In Anzeigeobjekten, die das Interaktionsmodell der InteractiveObject-Klasse erben, kann mithilfe von Ereignis-Listenern auf Tastaturereignisse reagiert werden. Beispielsweise können Sie einen Ereignis-Listener auf der Bühne platzieren, um auf Tastatureingaben zu warten und entsprechend zu reagieren. Im folgenden Code erfasst ein Ereignis-Listener einen Tastendruck. Zudem werden der Tastenname sowie die Tastencodeeigenschaften angezeigt.

```
function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (character code: " +
event.charCode + ")");
}
stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Einige Tasten, z. B. die Taste STRG, erzeugen Ereignisse, obwohl ihnen kein darstellbares Zeichen zugeordnet ist.

Im vorherigen Codebeispiel wurden mit dem Tastatur-Ereignis-Listener Tastatureingaben für die gesamte Bühne erfasst. Sie können auch einen Ereignis-Listener für bestimmte Anzeigeobjekte auf der Bühne programmieren. Dieser wird ausgelöst, wenn das entsprechende Objekt den Fokus hat.

Im folgenden Beispiel werden gedrückte Tasten nur dann im Bedienfeld „Ausgabe“ angezeigt, wenn die Benutzereingabe innerhalb der TextField-Instanz erfolgt. Beim Drücken der UMSCHALTTASTE wird die Rahmenfarbe der TextField-Instanz vorübergehend rot.

In diesem Code wird vorausgesetzt, dass auf der Bühne eine TextField-Instanz mit dem Namen `tf` positioniert ist.

```
tf.border = true;
tf.type = "input";
tf.addEventListener(KeyboardEvent.KEY_DOWN,reportKeyDown);
tf.addEventListener(KeyboardEvent.KEY_UP,reportKeyUp);

function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (key code: " +
event.keyCode + " character code: " + event.charCode + ")");
    if (event.keyCode == Keyboard.SHIFT) tf.borderColor = 0xFF0000;
}

function reportKeyUp(event:KeyboardEvent):void
{
    trace("Key Released: " + String.fromCharCode(event.charCode) + " (key code: " +
event.keyCode + " character code: " + event.charCode + ")");
    if (event.keyCode == Keyboard.SHIFT)
    {
        tf.borderColor = 0x000000;
    }
}
```

Wenn der Benutzer Text eingibt, meldet die TextField-Klasse auch ein `textInput`-Ereignis, auf das ebenfalls mit einem Ereignis-Listener gewartet werden kann. Weitere Informationen finden Sie unter „Erfassen von Texteingaben“ auf Seite 405.

Hinweis: *In der AIR-Laufzeit kann ein Tastaturereignis abgebrochen werden. In der Flash Player-Laufzeit kann ein Tastaturereignis nicht abgebrochen werden.*

Tastencodes und Zeichencodes

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können auf die Eigenschaften `keyCode` und `charCode` von Tastaturereignissen zugreifen, um die gedrückte Taste zu ermitteln und dann weitere Aktionen auszulösen. Die `keyCode`-Eigenschaft ist ein numerischer Wert, der dem Tastenwert einer Taste auf der Tastatur entspricht. Die `charCode`-Eigenschaft ist der numerische Wert dieser Taste im aktuellen Zeichensatz. (Der Standardzeichensatz ist UTF-8, in dem ASCII unterstützt wird.)

Der Hauptunterschied zwischen dem Tastencode und den Zeichenwerten besteht darin, dass ein Tastencodewert einer bestimmten Taste auf der Tastatur entspricht (die „1“ auf dem numerischen Ziffernblock unterscheidet sich von der „1“ in der obersten Tastenreihe; die Taste, die „1“ erzeugt, und die Taste, die „!“ erzeugt, ist jedoch dieselbe), während der Zeichenwert einem bestimmten Zeichen entspricht (die Zeichen „R“ und „r“ sind unterschiedlich).

Hinweis: *Einzelheiten zur Zuordnung von Tasten und Zeichencodewerten in ASCII finden Sie in der Beschreibung der flash.ui.Keyboard-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.*

Die Zuordnung zwischen Tasten und Tastencodes ist abhängig vom Gerät und vom Betriebssystem. Aus diesem Grund sollten Sie keine Tastenzuordnungen verwenden, um Aktionen auszulösen. Verwenden Sie stattdessen die vordefinierten Konstantenwerte der Keyboard-Klasse, um auf die entsprechenden `keyCode`-Eigenschaften zu verweisen. Verwenden Sie beispielsweise anstelle der Tastenzuordnung für die UMSCHALTTASTE die entsprechende Konstante `Keyboard.SHIFT` (wie im vorhergehenden Codebeispiel dargestellt).

KeyboardEvent-Reihenfolge

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie auch bei anderen Ereignissen wird die Tastaturereignisreihenfolge durch die Anzeigeobjekthierarchie und nicht durch die Reihenfolge bestimmt, in der `addEventListener()`-Methoden im Code zugewiesen werden.

Angenommen, Sie platzieren ein Textfeld mit dem Namen `tf` in einem Movieclip mit der Bezeichnung `container` und fügen für beide Instanzen einen Ereignis-Listener für Tastaturereignisse hinzu, wie im folgenden Beispiel dargestellt:

```
container.addEventListener(KeyboardEvent.KEY_DOWN,reportKeyDown);
container.tf.border = true;
container.tf.type = "input";
container.tf.addEventListener(KeyboardEvent.KEY_DOWN,reportKeyDown);

function reportKeyDown(event:KeyboardEvent):void
{
    trace(event.currentTarget.name + " hears key press: " + String.fromCharCode(event.charCode)
+ " (key code: " + event.keyCode + " character code: " + event.charCode + ")");
}
```

Da es sowohl für das Textfeld als auch für den übergeordneten Container einen Listener gibt, wird die `reportKeyDown()`-Funktion für jeden Tastendruck innerhalb der TextField-Instanz doppelt aufgerufen. Beachten Sie, dass bei jedem Tastendruck das Ereignis zuerst vom Textfeld und erst dann vom Movieclip `container` ausgelöst wird.

Tastaturereignisse werden zuerst im Betriebssystem und im Webbrowser und erst danach in Adobe Flash Player oder AIR verarbeitet. In Microsoft Internet Explorer führt das Drücken von STRG+W beispielsweise zum Schließen des Browserfensters, bevor in einer enthaltenen SWF-Datei überhaupt ein Tastaturereignis ausgelöst werden kann.

Verwenden der IME-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der IME-Klasse können Sie den Eingabemethoden-Editor (IME) des Betriebssystems über Flash Player oder AIR steuern.

Mit ActionScript können Sie ermitteln,

- ob ein IME auf dem Computer des Benutzers installiert ist (`Capabilities.hasIME`),
- ob der IME auf dem Computer des Benutzers aktiviert oder deaktiviert ist (`IME.enabled`),
- welchen Konvertierungsmodus der aktuelle IME verwendet (`IME.conversionMode`).

Sie können ein Eingabetextfeld mit einem bestimmten IME-Kontext verknüpfen. Wenn Sie zwischen Eingabefeldern wechseln, können Sie auch den IME beispielsweise zwischen Hiragana (Japanisch), Ziffern mit voller Breite, Ziffern mit halber Breite, direkter Eingabe usw. umschalten.

Benutzer können mithilfe eines IME ASCII-fremde Textzeichen aus Multibyte-Sprachen wie Chinesisch, Japanisch oder Koreanisch eingeben.

Weitere Informationen zur Verwendung von IMEs finden Sie in der Dokumentation des Betriebssystems, für das Sie die Anwendung entwickeln. Weitere Ressourcen finden Sie auf den folgenden Websites:

- http://www.msdn.microsoft.com/goglobal/
- http://developer.apple.com/library/mac/navigation/
- http://www.java.sun.com/

Hinweis: *Wenn ein IME auf dem Computer des Benutzers deaktiviert ist, treten mit Ausnahme von* `Capabilities.hasIME` *beim Aufrufen von „IME"-Methoden oder „IME"-Eigenschaften Fehler auf. Nach dem Aktivieren eines IME werden ActionScript-Aufrufe von „IME"-Methoden und „IME"-Eigenschaften wie erwartet durchgeführt. Wenn Sie beispielsweise einen IME für Japanisch verwenden, muss dieser aktiviert werden, bevor IME-Methoden oder IME-Eigenschaften aufgerufen werden können.*

Überprüfen, ob ein IME installiert und aktiviert ist

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Vor dem Aufrufen von IME-Methoden oder IME-Eigenschaften sollten Sie immer überprüfen, ob auf dem Computer des Benutzers ein IME installiert und aktiviert ist. Im folgenden Codebeispiel wird veranschaulicht, wie Sie vor dem Aufrufen von Methoden überprüfen können, ob ein IME installiert und aktiviert ist:

```
if (Capabilities.hasIME)
{
    if (IME.enabled)
    {
        trace("IME is installed and enabled.");
    }
    else
    {
        trace("IME is installed but not enabled. Please enable your IME and try again.");
    }
}
else
{
    trace("IME is not installed. Please install an IME and try again.");
}
```

Mit diesem Code wird zunächst mithilfe der `Capabilities.hasIME`-Eigenschaft überprüft, ob ein IME installiert ist. Wenn diese Eigenschaft auf `true` gesetzt ist, wird dann mit der `IME.enabled`-Eigenschaft überprüft, ob der IME aktiviert ist.

Ermitteln des derzeit aktivierten IME-Konvertierungsmodus

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Erstellen mehrsprachiger Anwendungen müssen Sie möglicherweise ermitteln, welcher Konvertierungsmodus aktiviert ist. Mit dem folgenden Codebeispiel wird veranschaulicht, wie überprüft wird, ob ein IME installiert ist, und wenn ja, welcher IME-Konvertierungsmodus aktiviert ist:

```
if (Capabilities.hasIME)
{
    switch (IME.conversionMode)
    {
        case IMEConversionMode.ALPHANUMERIC_FULL:
            tf.text = "Current conversion mode is alphanumeric (full-width).";
            break;
        case IMEConversionMode.ALPHANUMERIC_HALF:
            tf.text = "Current conversion mode is alphanumeric (half-width).";
            break;
        case IMEConversionMode.CHINESE:
            tf.text = "Current conversion mode is Chinese.";
            break;
        case IMEConversionMode.JAPANESE_HIRAGANA:
            tf.text = "Current conversion mode is Japananese Hiragana.";
            break;
        case IMEConversionMode.JAPANESE_KATAKANA_FULL:
            tf.text = "Current conversion mode is Japanese Katakana (full-width).";
            break;
        case IMEConversionMode.JAPANESE_KATAKANA_HALF:
            tf.text = "Current conversion mode is Japanese Katakana (half-width).";
            break;
        case IMEConversionMode.KOREAN:
            tf.text = "Current conversion mode is Korean.";
            break;
        default:
            tf.text = "Current conversion mode is " + IME.conversionMode + ".";
            break;
    }
}
else
{
    tf.text = "Please install an IME and try again.";
}
```

Mit diesem Code wird zunächst überprüft, ob ein IME installiert ist. Anschließend wird überprüft, welcher Konvertierungsmodus im aktuellen IME verwendet wird. Dazu wird die `IME.conversionMode`-Eigenschaft mit den einzelnen Konstanten der IMEConversionMode-Klasse verglichen.

Festlegen des IME-Konvertierungsmodus

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Ändern des Konvertierungsmodus des Benutzer-IME müssen Sie sicherstellen, dass der Code in einen `try..catch`-Block eingeschlossen ist, da das Festlegen eines Konvertierungsmodus mit der `conversionMode`-Eigenschaft einen Fehler auslösen kann, wenn der IME den Konvertierungsmodus nicht festlegen kann. Das folgende Beispiel veranschaulicht, wie Sie den `try..catch`-Block beim Festlegen der `IME.conversionMode`-Eigenschaft verwenden:

```
var statusText:TextField = new TextField;
statusText.autoSize = TextFieldAutoSize.LEFT;
addChild(statusText);
if (Capabilities.hasIME)
{
    try
    {
        IME.enabled = true;
        IME.conversionMode = IMEConversionMode.KOREAN;
        statusText.text = "Conversion mode is " + IME.conversionMode + ".";
    }
    catch (error:Error)
    {
        statusText.text = "Unable to set conversion mode.\n" + error.message;
    }
}
```

Mit diesem Code wird zunächst ein Textfeld erstellt, in dem eine Statusmeldung angezeigt werden kann. Wenn der IME installiert ist, wird dieser dann aktiviert und der Konvertierungsmodus auf Koreanisch gesetzt. Wenn auf dem Computer des Benutzers kein koreanischer IME installiert ist, wird in Flash Player oder AIR ein Fehler ausgelöst und vom `try..catch`-Block abgefangen. Der `try..catch`-Block zeigt die Fehlermeldung in dem zuvor erstellten Textfeld an.

Deaktivieren des IME für bestimmte Textfelder

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In einigen Fällen können Sie bei Bedarf den IME eines Benutzers deaktivieren, während der Benutzer Zeichen eingibt. Bei einem Textfeld, in dem nur numerische Zeichen eingegeben werden können, kann der IME beispielsweise deaktiviert werden, sodass die Dateneingabe nicht verlangsamt wird.

Im folgenden Codebeispiel wird veranschaulicht, wie Sie die Ereignisse `FocusEvent.FOCUS_IN` und `FocusEvent.FOCUS_OUT` überwachen und den IME des Benutzers entsprechend deaktivieren können:

```
var phoneTxt:TextField = new TextField();
var nameTxt:TextField = new TextField();

phoneTxt.type = TextFieldType.INPUT;
phoneTxt.addEventListener(FocusEvent.FOCUS_IN, focusInHandler);
phoneTxt.addEventListener(FocusEvent.FOCUS_OUT, focusOutHandler);
phoneTxt.restrict = "0-9";
phoneTxt.width = 100;
phoneTxt.height = 18;
phoneTxt.background = true;
phoneTxt.border = true;
addChild(phoneTxt);

nameField.type = TextFieldType.INPUT;
nameField.x = 120;
nameField.width = 100;
nameField.height = 18;
nameField.background = true;
nameField.border = true;
addChild(nameField);

function focusInHandler(event:FocusEvent):void
{
    if (Capabilities.hasIME)
    {
        IME.enabled = false;
    }
}
function focusOutHandler(event:FocusEvent):void
{
    if (Capabilities.hasIME)
    {
        IME.enabled = true;
    }
}
```

In diesem Beispiel werden die beiden Eingabetextfelder `phoneTxt` und `nameTxt` erstellt. Dann werden zwei Ereignis-Listener zum Textfeld `phoneTxt` hinzugefügt. Wenn der Benutzer den Fokus auf das Textfeld `phoneTxt` setzt, wird ein `FocusEvent.FOCUS_IN`-Ereignis ausgelöst und der IME deaktiviert. Wenn das Textfeld `phoneTxt` nicht mehr den Fokus hat, wird das `FocusEvent.FOCUS_OUT`-Ereignis ausgelöst und der IME wieder aktiviert.

Überwachen von IME-Kompositionsereignissen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

IME-Kompositionsereignisse werden ausgelöst, wenn ein Kompositionsstring festgelegt wird. Wenn der IME des Benutzers installiert und aktiviert ist und der Benutzer beispielsweise einen String auf Japanisch eingibt, wird das `IMEEvent.IME_COMPOSITION`-Ereignis ausgelöst, sobald der Benutzer den Kompositionsstring auswählt. Um das `IMEEvent.IME_COMPOSITION`-Ereignis überwachen zu können, müssen Sie einen Ereignis-Listener zur statischen `ime`-Eigenschaft der System-Klasse hinzufügen (`flash.system.System.ime.addEventListener(...)`), wie im folgenden Beispiel dargestellt:

```
var inputTxt:TextField;
var outputTxt:TextField;

inputTxt = new TextField();
inputTxt.type = TextFieldType.INPUT;
inputTxt.width = 200;
inputTxt.height = 18;
inputTxt.border = true;
inputTxt.background = true;
addChild(inputTxt);

outputTxt = new TextField();
outputTxt.autoSize = TextFieldAutoSize.LEFT;
outputTxt.y = 20;
addChild(outputTxt);

if (Capabilities.hasIME)
{
    IME.enabled = true;
    try
    {
        IME.conversionMode = IMEConversionMode.JAPANESE_HIRAGANA;
    }
    catch (error:Error)
    {
        outputTxt.text = "Unable to change IME.";
    }
    System.ime.addEventListener(IMEEvent.IME_COMPOSITION, imeCompositionHandler);
}
else
{
    outputTxt.text = "Please install IME and try again.";
}

function imeCompositionHandler(event:IMEEvent):void
{
    outputTxt.text = "you typed: " + event.text;
}
```

Mit diesem Code werden zwei Textfelder erstellt und dann zur Anzeigeliste hinzugefügt. Das erste Textfeld `inputTxt` ist ein Eingabetextfeld, in dem der Benutzer japanischen Text eingeben kann. Das zweite Textfeld `outputTxt` ist ein dynamisches Textfeld, in dem Fehlermeldungen angezeigt werden oder in dem der japanische String wiederholt wird, den der Benutzer im Textfeld `inputTxt` eingibt.

Virtuelle Tastaturen

Flash Player 10.2 und höher, AIR 2.6 und höher

Mobilgeräte wie Telefone und Tablets bieten oft eine virtuelle Softwaretastatur anstelle einer Hardwaretastatur. Die Klassen in der Flash-API bieten folgende Möglichkeiten:

- Erkennen, wann die virtuelle Tastatur eingeblendet und wann sie geschlossen wird.
- Verhindern, dass die Tastatur eingeblendet wird.

- Bestimmen des Bühnenbereichs, der von der virtuellen Tastatur verdeckt wird.
- Erstellen von interaktiven Objekten, die die Tastatur einblenden, wenn sie den Fokus erhalten. (Von AIR-Anwendungen unter iOS nicht unterstützt.)
- (Nur AIR) Deaktivieren des automatischen Schwenkverhaltens, damit die Anwendung die eigene Anzeige unter Berücksichtigung der Tastatur anpassen kann.

Steuern des Verhaltens von virtuellen Tastaturen

Die Laufzeit öffnet automatisch die virtuelle Tastatur, wenn der Benutzer in ein Textfeld oder auf ein speziell konfiguriertes, interaktives Objekt tippt. Wenn die Tastatur geöffnet wird, berücksichtigt die Laufzeit die Konventionen der nativen Plattform für das Verschieben und Ändern der Größe des Anwendungsinhalts, damit Benutzer den Text bei der Eingabe sehen können.

Wenn die Tastatur geöffnet wird, löst das Objekt, das den Fokus besitzt, die folgenden Ereignisse in der angegebenen Reihenfolge aus:

`softKeyboardActivating`-Ereignis – wird ausgelöst, unmittelbar bevor die Tastatur über der Bühne eingeblendet wird. Wenn Sie die `preventDefault()`-Methode des ausgelösten Ereignisobjekts aufrufen, wird die virtuelle Tastatur nicht geöffnet.

`softKeyboardActivate`-Ereignis – wird ausgelöst, nachdem die Verarbeitung des `softKeyboardActivating`-Ereignisses abgeschlossen ist. Wenn das Objekt, das den Fokus besitzt, dieses Ereignis auslöst, wurde die `softKeyboardRect`-Eigenschaft des Stage-Objekts aktualisiert, da nun ein Teil der Bühne von der virtuellen Tastatur verdeckt wird. Dieses Ereignis kann nicht abgebrochen werden.

Hinweis: *Wenn die Größe der Tastatur sich ändert, beispielsweise weil der Benutzer den Tastaturtyp ändert, löst das Objekt, das den Fokus besitzt, ein zweites softKeyboardActivate-Ereignis aus.*

`softKeyboardDeactivate`-Ereignis – wird ausgelöst, wenn die virtuelle Tastatur geschlossen wird. Dieses Ereignis kann nicht abgebrochen werden.

Im folgenden Beispiel werden zwei TextField-Objekte auf der Bühne hinzugefügt. Das obere TextField-Objekt verhindert, dass die Tastatur eingeblendet wird, wenn Sie auf das Feld tippen, und schließt die Tastatur, wenn sie bereits eingeblendet wurde. Das untere TextField-Objekt zeigt das Standardverhalten. Das Beispiel zeigt die Ereignisse der virtuellen Tastatur, die von beiden Textfeldern ausgelöst werden.

```
package
{
import flash.display.Sprite;
import flash.text.TextField;
import flash.text.TextFieldType;
import flash.events.SoftKeyboardEvent;
public class SoftKeyboardEventExample extends Sprite
{
    private var tf1:TextField = new TextField();
    private var tf2:TextField = new TextField();

    public function SoftKeyboardEventExample()
    {
        tf1.width = this.stage.stageWidth;
        tf1.type = TextFieldType.INPUT;
        tf1.border = true;
        this.addChild( tf1 );

        tf1.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATING, preventSoftKe
yboard );
        tf1.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE, preventSoftKe
yboard );
        tf1.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE, preventSoftKeyboard
);

        tf2.border = true;
        tf2.type = TextFieldType.INPUT;
        tf2.width = this.stage.stageWidth;
        tf2.y = tf1.y + tf1.height + 30;
        this.addChild( tf2 );

        tf2.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATING, allowSoftKeyboard );
        tf2.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE, allowSoftKeyboard );
        tf2.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE, allowSoftKeyboard);
    }

    private function preventSoftKeyboard( event:SoftKeyboardEvent ):void
    {
            event.preventDefault();
            this.stage.focus = null; //close the keyboard, if raised
            trace( "tf1 dispatched: " + event.type + " -- " + event.triggerType );
    }
    private function allowSoftKeyboard( event:SoftKeyboardEvent )    :void
    {
            trace( "tf2 dispatched: " + event.type + " -- " + event.triggerType );
    }
}
}
```

Unterstützung von virtuellen Tastaturen für interaktive Objekte

Flash Player 10.2 und höher, AIR 2.6 und höher (unter iOS nicht unterstützt)

Normalerweise wird die virtuelle Tastatur nur geöffnet, wenn der Benutzer auf ein TextField-Objekt tippt. Sie können eine Instanz der InteractiveObject-Klasse so konfigurieren, dass die virtuelle Tastatur geöffnet wird, wenn die Instanz den Fokus erhält.

Um eine InteractiveObject-Instanz so zu konfigurieren, dass sie die Softwaretastatur öffnet, setzen Sie ihre `needsSoftKeyboard`-Eigenschaft auf `true`. Die Softwaretastatur wird immer automatisch geöffnet, wenn das Objekt der focus-Eigenschaft der Bühne zugewiesen wird. Sie können die Tastatur auch einblenden, indem Sie die `requestSoftKeyboard()`-Methode der InteractiveObject-Instanz aufrufen.

Das folgende Beispiel zeigt, wie Sie eine InteractiveObject-Instanz als Texteingabefeld programmieren. Die im Beispiel gezeigte TextInput-Klasse stellt die `needsSoftKeyboard`-Eigenschaft so ein, dass die Tastatur bei Bedarf eingeblendet wird. Das Objekt wartet dann auf `keyDown`-Ereignisse und fügt das eingegebene Zeichen in das Feld ein.

Im Beispiel wird die Flash Text Engine verwendet, um eingegebenen Text anzufügen und anzuzeigen. Außerdem werden einige wichtige Ereignisse verarbeitet. Der Einfachheit halber ist im Beispiel kein Textfeld mit vollem Funktionsumfang implementiert.

```
package  {
    import flash.geom.Rectangle;
    import flash.display.Sprite;
    import flash.text.engine.TextElement;
    import flash.text.engine.TextBlock;
    import flash.events.MouseEvent;
    import flash.events.FocusEvent;
    import flash.events.KeyboardEvent;
    import flash.text.engine.TextLine;
    import flash.text.engine.ElementFormat;
    import flash.events.Event;

    public class TextInput extends Sprite
    {

        public var text:String = " ";
        public  var textSize:Number = 24;
        public var textColor:uint = 0x000000;
        private var _bounds:Rectangle = new Rectangle( 0, 0, 100, textSize );
        private var textElement: TextElement;
        private var textBlock:TextBlock = new  TextBlock();

        public function TextInput( text:String = "" )
        {
            this.text = text;
            this.scrollRect = _bounds;
            this.focusRect= false;

            //Enable keyboard support
            this.needsSoftKeyboard = true;
            this.addEventListener(MouseEvent.MOUSE_DOWN, onSelect);
            this.addEventListener(FocusEvent.FOCUS_IN, onFocusIn);
            this.addEventListener(FocusEvent.FOCUS_OUT, onFocusOut);

            //Setup text engine
            textElement = new TextElement( text, new ElementFormat( null, textSize, textColor ) );
```

```
            textBlock.content = textElement;
            var firstLine:TextLine = textBlock.createTextLine( null, _bounds.width - 8 );
            firstLine.x = 4;
            firstLine.y = 4 + firstLine.totalHeight;
            this.addChild( firstLine );

        }

        private function onSelect( event:MouseEvent ):void
        {
            stage.focus = this;
        }
        private function onFocusIn( event:FocusEvent ):void
        {
            this.addEventListener( KeyboardEvent.KEY_DOWN, onKey );
        }

        private function onFocusOut( event:FocusEvent ):void
        {
            this.removeEventListener( KeyboardEvent.KEY_UP, onKey );
        }

        private function onKey( event:KeyboardEvent ):void
        {
            textElement.replaceText( textElement.text.length, textElement.text.length,
String.fromCharCode( event.charCode ) );
            updateText();
        }
        public function set bounds( newBounds:Rectangle ):void
        {
            _bounds = newBounds.clone();
            drawBackground();
            updateText();
            this.scrollRect = _bounds;

            //force update to focus rect, if needed
            if( this.stage!= null && this.focusRect && this.stage.focus == this )
                this.stage.focus = this;
        }

        private function updateText():void
        {
            //clear text lines
            while( this.numChildren > 0 ) this.removeChildAt( 0 );

            //and recreate them
            var textLine:TextLine = textBlock.createTextLine( null, _bounds.width - 8);
            while ( textLine)
            {
                textLine.x = 4;
                if( textLine.previousLine != null )
                {
                    textLine.y = textLine.previousLine.y +
                                 textLine.previousLine.totalHeight + 2;
                }
                            else
```

```
                {
                    textLine.y = 4 + textLine.totalHeight;
                }
                this.addChild(textLine);
                textLine = textBlock.createTextLine(textLine, _bounds.width - 8 );
            }
        }

        private function drawBackground():void
        {
            //draw background and border for the field
            this.graphics.clear();
            this.graphics.beginFill( 0xededed );
            this.graphics.lineStyle( 1, 0x000000 );
            this.graphics.drawRect( _bounds.x + 2, _bounds.y + 2, _bounds.width - 4,
_bounds.height - 4);
            this.graphics.endFill();
        }
    }
}
```

Die folgende Hauptanwendungsklasse zeigt die Verwendung der TextInput-Klasse. Sie veranschaulicht auch, wie das Anwendungslayout verwaltet wird, wenn die Tastatur eingeblendet wird oder wenn die Ausrichtung des Geräts sich ändert. Die Hauptklasse erstellt ein TextInput-Objekt und stellt seine Begrenzungen so ein, dass das Objekt die Bühne ausfüllt. Die Klasse passt die Größe des TextInput-Objekts an, wenn die Softwaretastatur eingeblendet wird oder wenn die Größe der Bühne sich ändert. Die Klasse wartet auf Ereignisse zur Softwaretastatur vom TextInput-Objekt und auf resize-Ereignisse von der Bühne. Unabhängig von der Ursache des Ereignisses bestimmt die Anwendung den sichtbaren Bereich der Bühne und passt das Eingabesteuerelement so an, dass es diesen Bereich ausfüllt. In einer echten Anwendung wäre ein komplexerer Layoutalgorithmus erforderlich.

```
package  {

    import flash.display.MovieClip;
    import flash.events.SoftKeyboardEvent;
    import flash.geom.Rectangle;
    import flash.events.Event;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;

    public class CustomTextField extends MovieClip {

        private var customField:TextInput = new TextInput("Input text: ");

        public function CustomTextField() {
            this.stage.scaleMode = StageScaleMode.NO_SCALE;
            this.stage.align = StageAlign.TOP_LEFT;
            this.addChild( customField );
            customField.bounds = new Rectangle( 0, 0, this.stage.stageWidth,
this.stage.stageHeight );

            //track soft keyboard and stage resize events
            customField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE,
```

```
onDisplayAreaChange );
            customField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE,
onDisplayAreaChange );
            this.stage.addEventListener( Event.RESIZE, onDisplayAreaChange );
        }

        private function onDisplayAreaChange( event:Event ):void
        {
            //Fill the stage if possible, but avoid the area covered by a keyboard
            var desiredBounds = new Rectangle( 0, 0, this.stage.stageWidth,
this.stage.stageHeight );
            if( this.stage.stageHeight - this.stage.softKeyboardRect.height <
desiredBounds.height )
                desiredBounds.height = this.stage.stageHeight -
this.stage.softKeyboardRect.height;

            customField.bounds = desiredBounds;
        }
    }
}
```

Hinweis: *Die Bühne gibt nur resize-Ereignisse als Reaktion auf Ausrichtungsänderungen aus, wenn die* `scaleMode`*-Eigenschaft auf* `noScale` *gesetzt ist. In anderen Modi werden die Abmessungen der Bühne nicht geändert, sondern der Inhalt wird entsprechend skaliert.*

Verarbeiten von Änderungen an der Anwendungsanzeige

AIR 2.6 und höher

In AIR können Sie das Standardverhalten zum Schwenken und zum Ändern der Größe, das beim Einblenden der Softwaretastatur auftritt, deaktivieren, indem Sie das `softKeyboardBehavior`-Element im Anwendungsdeskriptor auf `none` einstellen:

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Wenn Sie das automatische Verhalten deaktivieren, ist die Anwendung dafür zuständig, die Anwendungsanzeige wie erforderlich anzupassen. Ein softKeyboardActivate-Ereignis wird beim Öffnen der Tastatur ausgelöst. Wenn das `softKeyboardActivate`-Ereignis ausgelöst wird, enthält die `softKeyboardRect`-Eigenschaft der Bühne die Abmessungen des Bereichs, der von der geöffneten Tastatur verdeckt wird. Ändern Sie die Position oder die Größe Ihres Inhalts anhand dieser Abmessungen, um sicherzustellen, dass der Inhalt richtig angezeigt wird, wenn die Tastatur geöffnet ist und der Benutzer Text eingibt. (Wenn die Tastatur geschlossen ist, betragen alle Abmessungen des softKeyboardRect-Rechtecks null.)

Beim Schließen der Tastatur wird ein `softKeyboardDeactivate`-Ereignis ausgelöst und Sie können wieder zur normalen Anwendungsanzeige zurückkehren.

```
package  {
    import flash.display.MovieClip;
    import flash.events.SoftKeyboardEvent;
    import flash.events.Event;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.display.InteractiveObject;
    import flash.text.TextFieldType;
    import flash.text.TextField;

    public class PanningExample extends MovieClip {

        private var textField:TextField = new TextField();

        public function PanningExample() {
            this.stage.scaleMode = StageScaleMode.NO_SCALE;
            this.stage.align = StageAlign.TOP_LEFT;

            textField.y = this.stage.stageHeight - 201;
            textField.width = this.stage.stageWidth;
            textField.height = 200;
            textField.type = TextFieldType.INPUT;
            textField.border = true;
            textField.wordWrap = true;
            textField.multiline = true;

            this.addChild( textField );

            //track soft keyboard and stage resize events
            textField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE,
onKeyboardChange );
            textField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE,
onKeyboardChange );
            this.stage.addEventListener( Event.RESIZE, onDisplayAreaChange );
        }

        private function onDisplayAreaChange( event:Event ):void
        {
            textField.y = this.stage.stageHeight - 201;
```

```
            textField.width = this.stage.stageWidth;
        }

        private function onKeyboardChange( event:SoftKeyboardEvent ):void
        {
            var field:InteractiveObject = textField;
            var offset:int = 0;

            //if the softkeyboard is open and the field is at least partially covered
            if( (this.stage.softKeyboardRect.y != 0) && (field.y + field.height >
this.stage.softKeyboardRect.y) )
                offset = field.y + field.height - this.stage.softKeyboardRect.y;

            //but don't push the top of the field above the top of the screen
            if( field.y - offset < 0 ) offset += field.y - offset;

            this.y = -offset;
        }
    }
}
```

Hinweis: *Unter Android sind die genauen Abmessungen der Tastatur in bestimmten Situationen nicht vom Betriebssystem abrufbar. Dies ist beispielsweise im Vollbildmodus der Fall. In diesen Fällen wird die Größe geschätzt. Bei der Ausrichtung im Querformat wird die native IME-Tastatur im Vollbildmodus für jede Texteingabe verwendet. Diese IME-Tastatur hat ein integriertes Texteingabefeld und verdeckt die ganze Bühne. Es gibt keine Möglichkeit, eine Tastatur im Querformat anzuzeigen, die nicht den ganzen Bildschirm ausfüllt.*

Kapitel 31: Mauseingabe

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ihre Anwendung kann durch das Erfassen von Mauseingaben und das Reagieren darauf Interaktivität erzeugen. Beachten Sie, dass in diesem Abschnitt vorausgesetzt wird, dass Sie mit dem Ereignismodell von ActionScript 3.0 vertraut sind. Weitere Informationen finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Einzelheiten zur Erkennung der verfügbaren Mausunterstützung (persistenter Cursor, Stift- oder Berührungseingabe) zur Laufzeit finden Sie unter „Erkennen von Eingabetypen“ auf Seite 594.

Verwandte Hilfethemen

flash.ui.Mouse

flash.events.MouseEvent

„Eingabe per Berührung, Multitouch und Gesten“ auf Seite 619

Erfassen von Mauseingaben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mausklicks erzeugen Mausereignisse, die zum Auslösen interaktiver Funktionen verwendet werden können. Sie können der Bühne einen Ereignis-Listener für sämtliche Mausereignisse innerhalb einer SWF-Datei hinzufügen. Sie können auch Ereignis-Listener für Objekte auf der Bühne hinzufügen, die von der InteractiveObject-Klasse erben (beispielsweise Sprite- oder MovieClip-Objekte). Diese Listener werden ausgelöst, wenn auf das entsprechende Objekt geklickt wird.

Wie bei Tastaturereignissen wird auch bei Mausereignissen die Aufstiegsphase durchlaufen. Da im folgenden Beispiel `square` ein der Bühne untergeordnetes Objekt ist, wird beim Klicken auf das Rechteck sowohl von der Sprite-Instanz `square` als auch vom Stage-Objekt ein Ereignis ausgelöst:

```
var square:Sprite = new Sprite();
square.graphics.beginFill(0xFF0000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.addEventListener(MouseEvent.CLICK, reportClick);
square.x =
square.y = 50;
addChild(square);

stage.addEventListener(MouseEvent.CLICK, reportClick);

function reportClick(event:MouseEvent):void
{
    trace(event.currentTarget.toString() + " dispatches MouseEvent. Local coords [" +
event.localX + "," + event.localY + "] Stage coords [" + event.stageX + "," + event.stageY + "]");
}
```

Beachten Sie, dass im obigen Beispiel das Mausereignis Positionsangaben für den Mausklick enthält. Die Eigenschaften `localX` und `localY` enthalten die Mausklickposition im letzten untergeordneten Objekt der Anzeigekette. Beispielsweise führt das Klicken auf die obere linke Ecke von `square` zur Anzeige der lokalen Koordinaten [0,0], da dies der Registrierungspunkt von `square` ist. Alternativ verweisen die Eigenschaften `stageX` und `stageY` auf die globalen Koordinaten des Mausklicks auf der Bühne. Für denselben Mausklick wird für diese Koordinaten [50,50] angezeigt, da `square` auf diese Koordinaten verschoben wurde. Beide Koordinatenpaare können nützlich sein, je nachdem, wie Sie auf Benutzerinteraktionen reagieren möchten.

Hinweis: *Im Vollbildmodus können Sie die Anwendung so konfigurieren, dass die Maussperre verwendet wird. Bei einer Maussperre ist der Cursor deaktiviert und unbegrenzte Mausbewegungen sind möglich. Weitere Informationen finden Sie unter „Verwenden des Vollbildmodus“ auf Seite 178.*

Das MouseEvent-Objekt enthält auch die booleschen Eigenschaften `altKey`, `ctrlKey` und `shiftKey`. Mithilfe dieser Eigenschaften können Sie überprüfen, ob zum Zeitpunkt des Mausklicks auch die Tasten ALT, STRG oder die UMSCHALTTASTE gedrückt wurden.

Ziehen von Sprite-Objekten auf der Bühne

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `startDrag()`-Methode der Sprite-Klasse können Sie Benutzern die Möglichkeit geben, Sprite-Objekte auf der Bühne zu ziehen. Im folgenden Code ist ein Beispiel hierfür dargestellt:

```
import flash.display.Sprite;
import flash.events.MouseEvent;

var circle:Sprite = new Sprite();
circle.graphics.beginFill(0xFFCC00);
circle.graphics.drawCircle(0, 0, 40);

var target1:Sprite = new Sprite();
target1.graphics.beginFill(0xCCFF00);
target1.graphics.drawRect(0, 0, 100, 100);
target1.name = "target1";

var target2:Sprite = new Sprite();
target2.graphics.beginFill(0xCCFF00);
target2.graphics.drawRect(0, 200, 100, 100);
target2.name = "target2";

addChild(target1);
addChild(target2);
addChild(circle);

circle.addEventListener(MouseEvent.MOUSE_DOWN, mouseDown)

function mouseDown(event:MouseEvent):void
{
    circle.startDrag();
}
circle.addEventListener(MouseEvent.MOUSE_UP, mouseReleased);

function mouseReleased(event:MouseEvent):void
{
    circle.stopDrag();
    trace(circle.dropTarget.name);
}
```

Weitere Einzelheiten finden Sie im Abschnitt zum Erstellen von Interaktionen durch Ziehen mit der Maus unter „Ändern der Position“ auf Seite 185.

Ziehen und Ablegen in AIR

In Adobe AIR können Sie Drag&Drop-Unterstützung aktivieren, damit Benutzer Daten in die Anwendung und aus der Anwendung ziehen können. Weitere Einzelheiten finden Sie unter „Ziehen und Ablegen in AIR“ auf Seite 648.

Anpassen des Mauszeigers

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der Mauszeiger kann ausgeblendet oder auch mit einem beliebigen Anzeigeobjekt auf der Bühne ausgetauscht werden. Wenn Sie den Mauszeiger ausblenden möchten, rufen Sie die `Mouse.hide()`-Methode auf. Passen Sie den Mauszeiger an, indem Sie `Mouse.hide()` aufrufen, auf das `MouseEvent.MOUSE_MOVE`-Ereignis für die Bühne warten und die Koordinaten eines Anzeigeobjekts (benutzerdefinierter Mauszeiger) auf die `stageX`-Eigenschaft und die `stageY`-Eigenschaft des Ereignisses setzen. Im folgenden Beispiel wird die grundlegende Vorgehensweise veranschaulicht:

```
var cursor:Sprite = new Sprite();
cursor.graphics.beginFill(0x000000);
cursor.graphics.drawCircle(0,0,20);
cursor.graphics.endFill();
addChild(cursor);

stage.addEventListener(MouseEvent.MOUSE_MOVE,redrawCursor);
Mouse.hide();

function redrawCursor(event:MouseEvent):void
{
    cursor.x = event.stageX;
    cursor.y = event.stageY;
}
```

Beispiel für Mauseingabe: WordSearch

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Dieses Beispiel veranschaulicht die Benutzerinteraktion durch Verarbeiten von Mausereignissen. Die Benutzer bilden möglichst viele Wörter aus einem zufällig ausgewählten Buchstabenraster. Dabei können die Buchstaben im Raster vertikal oder horizontal verbunden werden. Pro Wort dürfen Buchstaben jedoch nicht doppelt verwendet werden. In diesem Beispiel werden die folgenden Funktionen von ActionScript 3.0 veranschaulicht:

- Dynamisches Erstellen eines Komponentenrasters
- Reagieren auf Mausereignisse
- Führen eines Punktestands, der auf Benutzerinteraktionen beruht

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „WordSearch“ befinden sich im Ordner „Samples/WordSearch“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
WordSearch.as	Die Klasse mit den Hauptfunktionen der Anwendung.
WordSearch.fla oder WordSearch.mxml	Die Hauptanwendungsdatei für Flex (MXML) oder Flash (FLA).
dictionary.txt	Eine Datei, mit deren Hilfe ermittelt wird, ob die gebildeten Wörter gültig und korrekt geschrieben sind.

Laden eines Wörterbuchs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für ein Spiel, bei dem es um das Finden von Wörtern geht, wird ein Wörterbuch benötigt. Das Beispiel beinhaltet eine Textdatei mit dem Namen „dictionary.txt“, die eine durch Wagenrücklaufzeichen getrennte Liste von Wörtern enthält. Nach dem Erstellen eines Arrays mit der Bezeichnung `words` wird mit der `loadDictionary()`-Funktion diese Datei angefordert. Wenn sie erfolgreich geladen wurde, liegt die Datei zunächst als langer String vor. Mithilfe der `split()`-Methode können Sie diesen String in ein Array von Wörtern aufteilen. Dabei dienen Wagenrücklaufzeichen (Zeichencode 10) oder Zeilenvorschubzeichen (Zeichencode 13) zum Erkennen der Wortgrenze. Das Aufteilen erfolgt in der `dictionaryLoaded()`-Funktion:

```
words = dictionaryText.split(String.fromCharCode(13, 10));
```

Erstellen der Benutzeroberfläche

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Nachdem die Wörter gespeichert wurden, können Sie die Benutzeroberfläche einrichten. Erstellen Sie zwei Button-Instanzen: eine zum Absenden eines Wortes und eine zweite zum Löschen des gerade eingegebenen Wortes. In beiden Fällen müssen Sie auf die Benutzereingabe reagieren, indem Sie auf das von der Schaltfläche gesendete `MouseEvent.CLICK`-Ereignis warten und dann eine Funktion aufrufen. In der `setupUI()`-Funktion werden mit dem folgenden Code die Listener für die beiden Schaltflächen erstellt:

```
submitWordButton.addEventListener(MouseEvent.CLICK,submitWord);
clearWordButton.addEventListener(MouseEvent.CLICK,clearWord);
```

Erstellen des Spielbretts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Spielbrett ist ein Raster aus zufällig gewählten Buchstaben. In der `generateBoard()`-Funktion wird ein zweidimensionales Raster durch Verschachteln zweier Schleifen erstellt. In der ersten Schleife werden die Zeilen inkrementiert und in der zweiten die Gesamtanzahl der Spalten pro Zeile. Jede der durch diese Zeilen und Spalten erstellten Zellen enthält eine Schaltfläche, die einen Buchstaben auf dem Spielbrett repräsentiert.

```
private function generateBoard(startX:Number, startY:Number, totalRows:Number,
totalCols:Number, buttonSize:Number):void
{
    buttons = new Array();
    var colCounter:uint;
    var rowCounter:uint;
    for (rowCounter = 0; rowCounter < totalRows; rowCounter++)
    {
        for (colCounter = 0; colCounter < totalCols; colCounter++)
        {
            var b:Button = new Button();
            b.x = startX + (colCounter*buttonSize);
            b.y = startY + (rowCounter*buttonSize);
            b.addEventListener(MouseEvent.CLICK, letterClicked);
            b.label = getRandomLetter().toUpperCase();
            b.setSize(buttonSize,buttonSize);
            b.name = "buttonRow"+rowCounter+"Col"+colCounter;
            addChild(b);

            buttons.push(b);
        }
    }
}
```

Obwohl nur in einer Codezeile ein Listener für `MouseEvent.CLICK`-Ereignisse hinzugefügt wird, erfolgt die Zuweisung für jede Button-Instanz, da dies innerhalb einer `for`-Schleife geschieht. Außerdem ist jeder Schaltfläche ein Name zugeordnet, der aus der Zeilen- und Spaltenposition abgeleitet ist und eine einfache Möglichkeit bietet, im Code später auf die Zeile und Spalte der jeweiligen Schaltfläche zu verweisen.

Bilden von Wörtern aus Benutzereingaben

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wörter werden gebildet, indem im Raster horizontal oder vertikal angrenzende Buchstaben ausgewählt werden, ohne dabei einen Buchstaben doppelt zu verwenden. Mit jedem Mausklick wird ein Mausereignis erzeugt. Anschließend muss die Rechtschreibung des vom Benutzer eingegebenen Wortes geprüft werden, um sicherzustellen, dass es eine korrekte Fortsetzung der zuvor ausgewählten Buchstaben ist. Wenn dies nicht der Fall ist, wird das vorhergehende Wort gelöscht und erneut mit der Eingabe eines Wortes begonnen. Dies erfolgt mit der `isLegalContinuation()`-Methode.

```
private function isLegalContinuation(prevButton:Button, currButton:Button):Boolean
{
    var currButtonRow:Number = Number(currButton.name.charAt(currButton.name. indexOf("Row") +
3));
    var currButtonCol:Number = Number(currButton.name.charAt(currButton.name.indexOf("Col") +
3));
    var prevButtonRow:Number = Number(prevButton.name.charAt(prevButton.name.indexOf("Row") +
3));
    var prevButtonCol:Number = Number(prevButton.name.charAt(prevButton.name.indexOf("Col") +
3));

    return ((prevButtonCol == currButtonCol && Math.abs(prevButtonRow - currButtonRow) <= 1) ||
             (prevButtonRow == currButtonRow && Math.abs(prevButtonCol - currButtonCol) <= 1));
}
```

Mit den Methoden `charAt()` und `indexOf()` der String-Klasse werden die entsprechenden Zeilen und Spalten der aktuellen Schaltfläche und der Schaltfläche abgerufen, auf die zuvor geklickt wurde. Die `isLegalContinuation()`-Methode gibt den Wert `true` zurück, wenn sich eine Zeile oder Spalte im Vergleich zur vorher geklickten Schaltfläche nicht geändert hat und wenn sich die geänderte Spalte oder Zeile um den Wert 1 von der vorherigen unterscheidet. Wenn Sie die Spielregeln so ändern möchten, dass es zulässig ist, Wörter auch diagonal zu bilden, können Sie die Überprüfung auf jeweils eine nicht geänderte Zeile oder Spalte entfernen. Die entsprechende Codezeile hierfür lautet:

```
return (Math.abs(prevButtonRow - currButtonRow) <= 1) && Math.abs(prevButtonCol -
currButtonCol) <= 1));
```

Überprüfen der eingegebenen Wörter

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um den Programmcode für das Spiel abzuschließen, werden noch Mechanismen zum Überprüfen der übergebenen Wörter und zum Führen eines Punktestands benötigt. Diese befinden sich in der `searchForWord()`-Methode:

```
private function searchForWord(str:String):Number
{
    if (words && str)
    {
        var i:uint = 0
        for (i = 0; i < words.length; i++)
        {
            var thisWord:String = words[i];
            if (str == words[i])
            {
                return i;
            }
        }
        return -1;
    }
    else
    {
        trace("WARNING: cannot find words, or string supplied is null");
    }
    return -1;
}
```

Mit dieser Funktion werden in einer Schleife alle Wörter des Wörterbuchs überprüft. Wenn das Wort des Benutzers mit einem Wort im Wörterbuch übereinstimmt, wird die entsprechende Wörterbuchposition zurückgegeben. Mit der `submitWord()`-Methode wird anschließend die Antwort überprüft, und wenn es sich um eine gültige Position handelt, wird der Punktestand aktualisiert.

Anpassen des Beispiels

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Am Anfang der Klasse werden mehrere Konstanten definiert. Sie können das Spiel ändern, indem Sie die entsprechenden Werte bearbeiten. Beispielsweise können Sie die verfügbare Spielzeit verlängern, indem Sie den Wert der Variablen `TOTAL_TIME` erhöhen. Sie können auch den Wert der Variablen `PERCENT_VOWELS` geringfügig ändern, um die Wahrscheinlichkeit zu erhöhen, dass gültige Wörter gefunden werden.

Kapitel 32: Eingabe per Berührung, Multitouch und Gesten

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Die Funktionalität zur Verarbeitung von Berührungsereignissen auf der Flash-Plattform unterstützt die Eingabe über einen oder mehrere Kontaktpunkte auf berührungsempfindlichen Geräten. Außerdem können Flash-Laufzeiten Ereignisse verarbeiten, die mehrere Berührungspunkte mit einer Bewegung kombinieren, wodurch eine Geste entsteht. Die Flash-Laufzeiten können also zwei Eingabearten interpretieren:

Berührung Eingabe über einen Berührungspunkt, beispielsweise mit einem Finger oder Stift, auf einem berührungsempfindlichen Gerät. Einige Geräte unterstützen die Eingabe über mehrere Kontaktpunkte gleichzeitig mit einem Finger oder Stift.

Multitouch Eingabe über mehrere Kontaktpunkte gleichzeitig.

Geste Eingabe, die von einem Gerät oder Betriebssystem als Reaktion auf ein oder mehrere Berührungsereignisse interpretiert wird. Wenn der Benutzer beispielsweise eine Drehbewegung mit zwei Fingern gleichzeitig ausführt, interpretiert das Gerät oder Betriebssystem diese Berührungseingabe als Drehgeste. Einige Gesten werden mit einem Finger oder Berührungspunkt vorgenommen, andere erfordern mehrere Berührungspunkte. Das Gerät oder Betriebssystem bestimmt den Gestentyp, der zu der Eingabe gehört.

Sowohl bei der Berührungs- als auch bei der Gesteneingabe kann es sich je nach Endgerät um eine Multitouch-Eingabe handeln. ActionScript bietet APIs zur Verarbeitung von Berührungsereignissen, Gestenereignissen und einzeln verfolgten Berührungsereignissen für die Multitouch-Eingabe.

Hinweis: *Das Warten auf Berührungs- und Gestenereignisse kann sehr viel Verarbeitungsressourcen beanspruchen (ähnlich wie beim Rendern von mehreren Bildern pro Sekunde), je nach Computergerät und Betriebssystem. Wenn die zusätzliche Funktionalität von Berührungen oder Gesten nicht unbedingt erforderlich ist, empfiehlt es sich meist, stattdessen Mausereignisse zu verwenden. Wenn Sie nicht auf Berührungs- oder Gestenereignisse verzichten möchten, sollten Sie grafische Änderungen möglichst gering halten, besonders bei Ereignissen, die rasch ausgelöst werden können, wie beim Schwenken, Drehen oder Zoomen. Beispielsweise können Sie die Animation innerhalb einer Komponente unterbrechen, während der Benutzer ihre Größe mit einer Zoomgeste ändert.*

Verwandte Hilfethemen

flash.ui.Multitouch

flash.events.TouchEvent

flash.events.GestureEvent

flash.events.TransformGestureEvent

flash.events.GesturePhase

flash.events.PressAndTapGestureEvent

Paul Trani: Touch Events and Gestures on Mobile

Mike Jones: Virtual Game Controllers

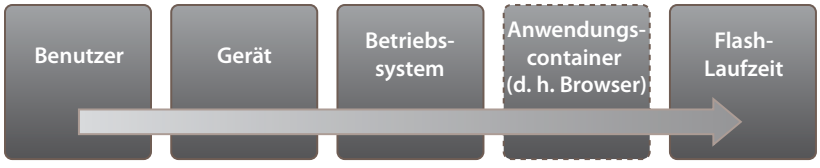
Grundlagen der Berührungseingabe

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Wenn die Flash-Plattform in einer Umgebung ausgeführt wird, die die Berührungseingabe unterstützt, können InteractiveObject-Instanzen auf Berührungsereignisse warten und Ereignisprozeduren aufrufen. Im Allgemeinen werden Berührungs-, Multitouch- und Gestenereignisse wie jedes andere Ereignis in ActionScript verarbeitet. (Grundlegende Informationen zur Ereignisverarbeitung in ActionScript finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.)

Doch damit die Flash-Laufzeit Berührungen oder Gesten interpretieren kann, muss die Hardware- und Software-Umgebung, in der die Laufzeit ausgeführt wird, die Eingabe per Berührung oder Multitouch unterstützen. Unter „Erkennen von Eingabetypen“ auf Seite 594 finden Sie eine Vergleichsübersicht über die verschiedenen Touchscreen-Typen. Wenn die Laufzeit in einer Containeranwendung ausgeführt wird (wie beispielsweise in einem Browser), ist außerdem zu beachten, dass dieser Container die Eingabe an die Laufzeit übergibt. In manchen Fällen bieten die Hardware und das Betriebssystem zwar Multitouch-Unterstützung, doch der Browser, in dem die Flash-Laufzeit ausgeführt wird, interpretiert die Eingabe und gibt sie nicht an die Laufzeit weiter. Möglicherweise wird die Eingabe vom Browser auch vollständig ignoriert.

Die folgende Abbildung zeigt den Verlauf der Eingabe vom Benutzer zur Laufzeit:



Verlauf der Eingabe vom Benutzer zur Laufzeit der Flash-Plattform

Die ActionScript-API zur Entwicklung von Anwendungen mit Berührungserkennung umfasst Klassen, Methoden und Eigenschaften, mit denen ermittelt werden kann, ob die Eingabe per Berührung oder Multitouch in der Laufzeitumgebung unterstützt wird. Zu diesem Zweck verwenden Sie die „Discovery API“ (Erkennungs-API) für die Verarbeitung von Berührungsereignissen.

Wichtige Konzepte und Begriffe

In der folgenden Liste werden wichtige Begriffe erläutert, die beim Erstellen von Anwendungen mit Berührungsereignisverarbeitung von Bedeutung sind:

Discovery API Die Methoden und Eigenschaften, mit denen ermittelt werden kann, ob die Laufzeitumgebung Berührungsereignisse und verschiedene Eingabemodi unterstützt.

Berührungsereignis Eine Eingabeaktion, die auf einem berührungsempfindlichen Gerät über einen einzelnen Kontaktpunkt vorgenommen wird.

Berührungspunkt Der Kontaktpunkt bei einem einzelnen Berührungsereignis. Auch wenn ein Gerät die Gesteneingabe nicht unterstützt, bietet es möglicherweise Unterstützung für mehrere Berührungspunkte gleichzeitig.

Berührungssequenz Die Ereignisserie, die die Abfolge einer einzelnen Berührung darstellt. Dazu zählen der Anfang, keine oder mehrere Bewegungen und das Ende der Berührungssequenz.

Multitouch-Ereignis Eine Eingabeaktion, die auf einem berührungsempfindlichen Gerät über mehrere Kontaktpunkte vorgenommen wird (etwa mit mehreren Fingern).

Gestenereignis Eine Eingabeaktion, die auf einem berührungsempfindlichen Gerät über eine komplexe Bewegung vorgenommen wird. Eine Geste kann beispielsweise ausgeführt werden, indem der Benutzer den Bildschirm mit zwei Fingern berührt und die Finger dann gleichzeitig kreisförmig bewegt, um eine Drehung anzuzeigen.

Phasen Bestimmte Zeitpunkte im Ereignisablauf (wie Beginn und Ende).

Stift Ein Instrument zur Interaktion mit einem berührungsempfindlichen Bildschirm. Mithilfe eines Stifts lässt sich die Berührung präziser steuern als mit einem Finger. Manche Geräte erkennen nur die Eingabe über einen bestimmten Stift. Geräte, die die Stifteingabe erkennen, können nicht unbedingt auch mehrere Kontaktpunkte gleichzeitig oder die Eingabe per Finger erkennen.

Drücken und Tippen Eine bestimmte Art von Multitouch-Eingabegeste, bei der der Benutzer mit einem Finger auf das berührungsempfindliche Gerät drückt und dann mit einem anderen Finger oder Zeigegerät darauf tippt. Mit dieser Geste wird häufig das Klicken mit der rechten Maustaste in Multitouch-Anwendungen simuliert.

Struktur der API für die Berührungseingabe

Mit der ActionScript-API für die Berührungseingabe kann berücksichtigt werden, dass die Verarbeitung der Berührungseingabe sich nach der Hardware- und Software-Umgebung der Flash-Laufzeit richtet. Die API für die Berührungseingabe ist schwerpunktmäßig für drei Aspekte der Entwicklung von berührungsempfindlichen Anwendungen vorgesehen: Erkennung, Ereignisse und Phasen. Durch eine Koordination dieser API-Aspekte können Sie ein vorhersehbares Anwendungsverhalten mit angemessenen Reaktionen für den Benutzer entwickeln, auch wenn bei der Entwicklung der Anwendung noch nicht bekannt ist, auf welchem Zielgerät sie ausgeführt wird.

Erkennung

Mit der Discovery API zur Erkennung kann die Hardware- und Software-Umgebung zur Laufzeit überprüft werden. Die von der Laufzeit angegebenen Werte bestimmen, welche Berührungseingabe in der Flash-Laufzeit im aktuellen Kontext verfügbar ist. Richten Sie Ihre Anwendung mithilfe der Erkennungseigenschaften und -methoden auch so ein, dass sie auf Mausereignisse reagieren kann (anstelle von Berührungsereignissen, falls eine bestimmte Berührungseingabe nicht von der Umgebung unterstützt wird). Weitere Informationen finden Sie unter „Erkennung der Berührungsunterstützung“ auf Seite 622.

Ereignisse

ActionScript verarbeitet Ereignisse für die Berührungseingabe wie jedes andere Ereignis mithilfe von Ereignis-Listenern und Ereignisprozeduren. Bei der Verarbeitung von Ereignissen für die Berührungseingabe müssen jedoch auch folgende Aspekte berücksichtigt werden:

- Berührungen können vom Gerät oder Betriebssystem entweder als Sequenz aus einzelnen Berührungen oder insgesamt als Geste interpretiert werden.
- Bei einer einzelnen Berührung eines berührungsempfindlichen Geräts (mit einem Finger, Stift oder Zeigegerät) wird auch immer ein Mausereignis ausgelöst. Das Mausereignis kann mit den Ereignistypen der MouseEvent-Klasse verarbeitet werden. Sie können Ihre Anwendung aber auch so gestalten, dass sie entweder nur auf Berührungsereignisse oder auf beide Ereignistypen reagiert.
- Eine Anwendung kann auf mehrere gleichzeitig ausgeführte Berührungsereignisse reagieren und jedes dieser Ereignisse separat verarbeiten.

Normalerweise verwenden Sie die Discovery API für eine bedingte Verarbeitung der Ereignisse in der Anwendung und um festzulegen, wie die Ereignisse verarbeitet werden. Sobald die Anwendung die Laufzeitumgebung kennt, kann sie die passende Ereignisprozedur aufrufen oder das richtige Ereignisobjekt bestimmen, wenn der Benutzer mit der Anwendung interagiert. Wenn die Anwendung feststellt, dass eine bestimmte Eingabeart in der aktuellen Umgebung nicht unterstützt wird, kann sie dem Benutzer eine Alternative oder entsprechende Informationen bereitstellen. Weitere Informationen finden Sie unter „Verarbeitung von Berührungsereignissen“ auf Seite 623 und „Verarbeitung von Gestenereignissen“ auf Seite 628.

Phasen

Bei Anwendungen mit Berührungs- und Multitouch-Eingabe enthalten die Berührungsereignisobjekte Eigenschaften, mit denen die Phasen der Benutzerinteraktion verfolgt werden. Erstellen Sie ActionScript-Code zur Verarbeitung der Benutzereingabephasen wie „Beginn“, „Aktualisierung“ oder „Ende“, um dem Benutzer Feedback zu geben. Implementieren Sie Reaktionen auf Ereignisphasen, damit visuelle Objekte sich ändern, wenn der Benutzer sie berührt und den Berührungspunkt auf dem Bildschirm verschiebt. Sie können die Phasen auch verwenden, um bestimmte Eigenschaften einer Geste zu verfolgen, während die Geste durchgeführt wird.

Bei Berührungspunktereignissen sollte verfolgt werden, wie lange der Benutzer ein bestimmtes interaktives Objekt berührt. Eine Anwendung kann die Phasen mehrerer gleichzeitiger Berührungspunkte einzeln verfolgen und entsprechend verarbeiten.

Bei einer Geste sollten spezifische Informationen über die Transformation der Geste interpretiert werden. Verfolgen Sie die Koordinaten von einem oder mehreren Kontaktpunkten bei der Bewegung über den Bildschirm.

Erkennung der Berührungsunterstützung

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Verwenden Sie die Eigenschaften der Multitouch-Klasse, um den Umfang der von Ihrer Anwendung verarbeiteten Berührungseingabe festzulegen. Überprüfen Sie dann die Umgebung, um sicherzustellen, dass die von ActionScript verarbeiteten Ereignisse unterstützt werden. Besonders wichtig ist es, zunächst den Typ der Berührungseingabe für Ihre Anwendung festzulegen. Folgende Optionen sind verfügbar: Berührungspunkt, Geste oder keine Berührungseingabe (dabei werden alle Berührungseingaben als Mausklicks interpretiert und es werden nur Prozeduren für Mausereignisse verwendet). Verwenden Sie dann die Eigenschaften und Methoden der Multitouch-Klasse, um sicherzustellen, dass die Berührungseingabe der Anwendung in der Laufzeitumgebung unterstützt wird. Überprüfen Sie die Laufzeitumgebung hinsichtlich der unterstützten Berührungseingabetypen (beispielsweise ob Gesten interpretiert werden können) und implementieren Sie entsprechende Reaktionen.

Hinweis: *Die Eigenschaften der Multitouch-Klasse sind statische Eigenschaften, die nicht zu Instanzen einer Klasse gehören. Verwenden Sie sie mit der Syntax „Multitouch.Eigenschaft“, wie zum Beispiel:*

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

Festlegen des Eingabetyps

Die Flash-Laufzeit muss den Typ der zu interpretierenden Berührungseingabe kennen, da ein Berührungsereignis aus zahlreichen Elementen oder Phasen bestehen kann. Löst die Laufzeit schon ein Berührungsereignis aus, wenn ein berührungsempfindlicher Bildschirm lediglich mit einem Finger berührt wird? Oder wartet die Laufzeit auf eine Geste? Oder wird die Berührung als mouse-down-Ereignis (gedrückte Maustaste) verfolgt? Eine Anwendung, die die Berührungseingabe unterstützt, muss die Typen der Berührungsereignisse festlegen, die in der Flash-Laufzeit verarbeitet werden. Verwenden Sie die `Multitouch.inputMode`-Eigenschaft, um den Typ der Berührungseingabe für die Laufzeit festzulegen. Drei Eingabemodi sind verfügbar:

Keine Es ist keine besondere Verarbeitung für Berührungsereignisse vorgesehen. Stellen Sie `Multitouch.inputMode=MultitouchInputMode.NONE` ein und verarbeiten Sie die Eingabe über die MouseEvent-Klasse.

Einzelne Berührungspunkte Jede Berührungseingabe wird einzeln interpretiert und alle Berührungspunkte können verfolgt und verarbeitet werden. Stellen Sie `Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT` ein und verarbeiten Sie die Eingabe über die TouchEvent-Klasse.

Gesteneingabe Das Gerät oder Betriebssystem interpretiert die Eingabe als komplexe Fingerbewegung über den Bildschirm. Das Gerät oder Betriebssystem weist die Bewegung insgesamt einem einzelnen Gesteneingabe-Ereignis zu. Stellen Sie `Multitouch.inputMode=MultitouchInputMode.GESTURE` ein und verarbeiten Sie die Eingabe über die TransformGestureEvent-, PressAndTapGestureEvent- oder GestureEvent-Klassen.

Unter „Verarbeitung von Berührungsereignissen“ auf Seite 623 finden Sie ein Beispiel dafür, wie die `Multitouch.inputMode`-Eigenschaft verwendet wird, um vor der Verarbeitung eines Berührungsereignisses den Eingabetyp festzulegen.

Überprüfen der Unterstützung für die Berührungseingabe

Mit anderen Eigenschaften der Multitouch-Klasse können Sie Ihre Anwendung genau an die Berührungsunterstützung der aktuellen Umgebung anpassen. Die Flash-Laufzeit gibt Werte für die Anzahl der gleichzeitig zulässigen Berührungspunkte oder der verfügbaren Gesten an. Wenn die Laufzeitumgebung die Berührungsereignisse Ihrer Anwendung nicht verarbeiten kann, stellen Sie den Benutzern Alternativen bereit, beispielsweise Mausereignisverarbeitung oder Informationen dazu, welche Funktionsmerkmale in der aktuellen Umgebung verfügbar sind oder nicht.

Sie können auch die API für Tastatur-, Berührungs- und Mausunterstützung verwenden; siehe „Erkennen von Eingabetypen“ auf Seite 594.

Weitere Informationen zu Kompatibilitätstests finden Sie unter „Fehlerbehebung“ auf Seite 632.

Verarbeitung von Berührungsereignissen

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Einfache Berührungsereignisse werden genau wie alle anderen Ereignisse (beispielsweise Mausereignisse) in ActionScript verarbeitet. Sie können auf eine Serie von Berührungsereignissen warten, die durch die Ereignistypkonstanten der TouchEvent-Klasse definiert sind.

Hinweis: *Bei der Eingabe über mehrere Berührungspunkte (wie bei der Berührung eines Geräts mit mehreren Fingern) löst der erste Kontaktpunkt ein Mausereignis und ein Berührungsereignis aus.*

Verarbeiten eines einfachen Berührungsereignisses:

1 Stellen Sie die `flash.ui.Multitouch.inputMode`-Eigenschaft auf `MultitouchInputMode.TOUCH_POINT` ein, damit Ihre Anwendung Berührungsereignisse verarbeiten kann.

2 Fügen Sie einen Ereignis-Listener an eine Instanz einer Klasse an, die Eigenschaften von der InteractiveObject-Klasse erbt, wie beispielsweise Sprite oder TextField.

3 Geben Sie den Typ des zu verarbeitenden Berührungsereignisses an.

4 Rufen Sie eine Ereignisprozedurfunktion auf, die auf das Ereignis reagiert.

Mit dem folgenden Code wird beispielsweise eine Meldung eingeblendet, wenn der Benutzer auf einem berührungsempfindlichen Bildschirm auf das mySprite-Quadrat tippt.

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taphandler);

function taphandler(evt:TouchEvent): void {
    myTextField.text = "I've been tapped";
    myTextField.y = 50;
    addChild(myTextField);
}
```

Eigenschaften von Berührungsereignissen

Wenn ein Ereignis auftritt, wird ein Ereignisobjekt erstellt. Das TouchEvent-Objekt enthält Informationen zur Position und zu den Bedingungen des Berührungsereignisses. Sie können diese Informationen über die Eigenschaften des Ereignisobjekts abrufen.

Der folgende Code erstellt beispielsweise das TouchEvent-Objekt *evt* und zeigt dann die `stageX`-Eigenschaft des Ereignisobjekts (die x-Koordinate des Punkts auf der Bühne, der berührt wurde) im Textfeld an:

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taphandler);

function taphandler(evt:TouchEvent): void {
myTextField.text = evt.stageX.toString;
myTextField.y = 50;
addChild(myTextField);
}
```

Im Eintrag zur TouchEvent-Klasse wird beschrieben, welche Eigenschaften über das Ereignisobjekt zur Verfügung stehen.

Hinweis: *Nicht alle TouchEvent-Ereignisse werden in allen Laufzeitumgebungen unterstützt. So können beispielsweise nicht alle berührungsempfindlichen Geräte die Stärke des Drucks erkennen, die der Benutzer auf den Touchscreen ausübt. Die `TouchEvent.pressure`-Eigenschaft wird deshalb auf diesen Geräten nicht unterstützt. Überprüfen Sie, ob bestimmte Eigenschaften unterstützt werden, um sicherzustellen, dass Ihre Anwendung funktioniert. Weitere Informationen finden Sie unter „Fehlerbehebung“ auf Seite 632.*

Phasen eines Berührungsereignisses

Verfolgen Sie Berührungsereignisse während der verschiedenen Phasen innerhalb und außerhalb eines InteractiveObject, genau wie bei Mausereignissen. Verfolgen Sie Berührungsereignisse auch am Anfang, im Verlauf und am Ende einer Berührungsinteraktion. Die TouchEvent-Klasse bietet Werte zur Verarbeitung der Ereignisse `touchBegin`, `touchMove` und `touchEnd`.

Beispielsweise können Sie Benutzern mithilfe der Ereignisse `touchBegin`, `touchMove` und `touchEnd` visuelles Feedback bereitstellen, wenn sie ein Anzeigeobjekt berühren und verschieben:

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);
var myTextField:TextField = new TextField();
myTextField.width = 200;
myTextField.height = 20;
addChild(myTextField);

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
function onTouchBegin(event:TouchEvent) {
    myTextField.text = "touch begin" + event.touchPointID;
}
function onTouchMove(event:TouchEvent) {
    myTextField.text = "touch move" + event.touchPointID;
}
function onTouchEnd(event:TouchEvent) {
    myTextField.text = "touch end" + event.touchPointID;
}
```

Hinweis: *Der Listener für die anfängliche Berührung ist an mySprite angefügt, bei den Listenern für das Verschieben und für das Ende des Berührungsereignisses ist dies jedoch nicht der Fall. Wenn der Benutzer seinen Finger oder das Zeigegerät vor das Anzeigeobjekt verschiebt, wartet die Bühne weiterhin auf das Berührungsereignis.*

Berührungspunkt-ID

Die `TouchEvent.touchPointID`-Eigenschaft ist beim Schreiben von Anwendungen, die auf die Berührungseingabe reagieren, ein wichtiger Bestandteil. Die Flash-Laufzeit weist jedem Berührungspunkt einen eindeutigen `touchPointID`-Wert zu. Wenn eine Anwendung auf die Phasen oder die Bewegung einer Berührungseingabe reagiert, sollten Sie vor der Verarbeitung des Ereignisses immer die `touchPointID` überprüfen. Die Ziehmethoden für die Berührungseingabe der Sprite-Klasse verwenden die `touchPointID`-Eigenschaft als Parameter, damit die richtige Eingabeinstanz verarbeitet wird. Die `touchPointID`-Eigenschaft gewährleistet, dass eine Ereignisprozedur auf den richtigen Berührungspunkt reagiert. Andernfalls reagiert die Ereignisprozedur auf alle Instanzen eines Berührungsereignistyps (wie beispielsweise alle `touchMove`-Ereignisse) auf dem Gerät, wodurch ein unvorhersehbares Verhalten auftritt. Diese Eigenschaft ist besonders wichtig, wenn der Benutzer Objekte zieht.

Verwenden Sie die `touchPointID`-Eigenschaft zur Verarbeitung einer ganzen Berührungssequenz. Eine Berührungssequenz hat ein `touchBegin`-Ereignis, keine oder mehrere `touchMove`-Ereignisse und ein `touchEnd`-Ereignis. Alle diese Ereignisse haben denselben `touchPointID`-Wert.

Im folgenden Beispiel wird eine *touchMoveID*-Variable eingerichtet, um auf den richtigen `touchPointID`-Wert zu prüfen, bevor auf ein Berührungsverschiebungs-Ereignis reagiert wird. Andernfalls lösen auch andere Berührungseingaben diese Ereignisprozedur aus. Beachten Sie, dass sich die Listener für die Verschiebungs- und Endphasen auf der Bühne, nicht im Anzeigeobjekt befinden. Die Bühne wartet auf die Bewegungs- und Endphasen für den Fall, dass die Berührung über die Begrenzungen des Anzeigeobjekts hinausreicht.

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);
var myTextField:TextField = new TextField();
addChild(myTextField);
myTextField.width = 200;
myTextField.height = 20;
var touchMoveID:int = 0;

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
function onTouchBegin(event:TouchEvent) {
    if(touchMoveID != 0) {
        myTextField.text = "already moving. ignoring new touch";
        return;
    }
    touchMoveID = event.touchPointID;

    myTextField.text = "touch begin" + event.touchPointID;
    stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
    stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
```

```
}
function onTouchMove(event:TouchEvent) {
    if(event.touchPointID != touchMoveID) {
        myTextField.text = "ignoring unrelated touch";
        return;
    }
    mySprite.x = event.stageX;
    mySprite.y = event.stageY;
    myTextField.text = "touch move" + event.touchPointID;
}
function onTouchEnd(event:TouchEvent) {
    if(event.touchPointID != touchMoveID) {
        myTextField.text = "ignoring unrelated touch end";
        return;
    }
    touchMoveID = 0;
    stage.removeEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
    stage.removeEventListener(TouchEvent.TOUCH_END, onTouchEnd);
    myTextField.text = "touch end" + event.touchPointID;
}
```

Berühren und Ziehen

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Zwei Methoden wurden der Sprite-Klasse hinzugefügt, um zusätzliche Unterstützung für berührungsempfindliche Anwendungen zu bieten, die die Eingabe über Berührungspunkte unterstützen: `Sprite.startTouchDrag()` und `Sprite.stopTouchDrag()`. Diese Methoden verhalten sich genauso wie `Sprite.startDrag()` und `Sprite.stopDrag()` bei Mausereignissen. Beachten Sie jedoch, dass die Methoden `Sprite.startTouchDrag()` und `Sprite.stopTouchDrag()` beide `touchPointID`-Werte als Parameter verwenden.

Die Laufzeit weist dem Ereignisobjekt den `touchPointID`-Wert für ein Berührungsereignis zu. Verwenden Sie diesen Wert, um auf einen bestimmten Berührungspunkt zu reagieren, falls die Umgebung mehrere Berührungspunkte gleichzeitig unterstützt (auch wenn keine Gesten verarbeitet werden). Weitere Informationen zur `touchPointID`-Eigenschaft finden Sie unter „Berührungspunkt-ID“ auf Seite 626.

Der folgende Code zeigt je eine einfache Ereignisprozedur für den Anfang und das Ende einer Ziehbewegung bei einem Berührungsereignis. Die Variable *bg* ist ein Anzeigeobjekt, das *mySprite* enthält:

```
mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
mySprite.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);

function onTouchBegin(e:TouchEvent) {
    e.target.startTouchDrag(e.touchPointID, false, bg.getRect(this));
    trace("touch begin");
}

function onTouchEnd(e:TouchEvent) {
    e.target.stopTouchDrag(e.touchPointID);
    trace("touch end");
}
```

Der folgende Code zeigt ein etwas komplexeres Beispiel, in dem das Ziehen mit Berührungsereignisphasen kombiniert wird:

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
mySprite.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
mySprite.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);

function onTouchBegin(evt:TouchEvent) {
    evt.target.startTouchDrag(evt.touchPointID);
    evt.target.scaleX *= 1.5;
    evt.target.scaleY *= 1.5;
}

function onTouchMove(evt:TouchEvent) {
    evt.target.alpha = 0.5;
}

function onTouchEnd(evt:TouchEvent) {
    evt.target.stopTouchDrag(evt.touchPointID);
    evt.target.width = 40;
    evt.target.height = 40;
    evt.target.alpha = 1;
}
```

Verarbeitung von Gestenereignissen

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Gestenereignisse werden genauso verarbeitet wie einfache Berührungsereignisse. Sie können auf eine Serie von Gestenereignissen warten, die durch Ereignistypkonstanten in den Klassen TransformGestureEvent, GestureEvent und PressAndTapGestureEvent definiert sind.

Verarbeiten eines Gestenberührungsereignisses:

1. Stellen Sie die `flash.ui.Multitouch.inputMode`-Eigenschaft auf `MultitouchInputMode.GESTURE` ein, damit Ihre Anwendung die Gesteneingabe verarbeiten kann.
2. Fügen Sie einen Ereignis-Listener an eine Instanz einer Klasse an, die Eigenschaften von der InteractiveObject-Klasse erbt, wie beispielsweise Sprite oder TextField.
3. Geben Sie den Typ des zu verarbeitenden Gestenereignisses an.
4. Rufen Sie eine Ereignisprozedurfunktion auf, die auf das Ereignis reagiert.

Mit dem folgenden Code wird beispielsweise eine Meldung eingeblendet, wenn der Benutzer auf einem berührungsempfindlichen Bildschirm eine Swipe-Bewegung über das Quadrat auf *mySprite* durchführt:

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TransformGestureEvent.GESTURE_SWIPE, swipehandler);

function swipehandler(evt:TransformGestureEvent): void {
myTextField.text = "I've been swiped";
myTextField.y = 50;
addChild(myTextField);
}
```

Tippereignisse mit zwei Fingern werden genauso verarbeitet, verwenden aber die GestureEvent-Klasse:

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(GestureEvent.GESTURE_TWO_FINGER_TAP, taphandler);

function taphandler(evt:GestureEvent): void {
myTextField.text = "I've been two-finger tapped";
myTextField.y = 50;
addChild(myTextField);
}
```

Ereignisse, die aus Drücken und Tippen bestehen, werden ebenfalls genauso verarbeitet, sie verwenden jedoch die PressAndTapGestureEvent-Klasse:

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(PressAndTapGestureEvent.GESTURE_PRESS_AND_TAP, taphandler);

function taphandler(evt:PressAndTapGestureEvent): void {
myTextField.text = "I've been press-and-tapped";
myTextField.y = 50;
addChild(myTextField);
}
```

Hinweis: *Nicht alle GestureEvent-, TransformGestureEvent- und PressAndTapGestureEvent-Ereignistypen werden in allen Laufzeitumgebungen unterstützt. Beispielsweise können nicht alle berührungsempfindlichen Geräte eine Swipe-Bewegung mit mehreren Fingern erkennen. Die InteractiveObject-Ereignisse* `gestureSwipe` *werden deshalb auf diesen Geräten nicht unterstützt. Überprüfen Sie, ob bestimmte Ereignisse unterstützt werden, um sicherzustellen, dass Ihre Anwendung funktioniert. Weitere Informationen finden Sie unter „Fehlerbehebung“ auf Seite 632.*

Eigenschaften von Gestenereignissen

Für Gestenereignisse sind weniger Eigenschaften verfügbar als für einfache Berührungsereignisse. Der Zugriff erfolgt auf dieselbe Weise über das Ereignisobjekt in der Ereignisprozedurfunktion.

Mit dem folgenden Code wird beispielsweise *mySprite* gedreht, wenn der Benutzer eine Drehgeste ausführt. Das Textfeld zeigt den Drehbetrag seit der letzten Geste (drehen Sie das Objekt beim Testen dieses Codes mehrmals, um zu sehen, wie die Werte sich ändern):

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var mySpriteCon:Sprite = new Sprite();
var myTextField:TextField = new TextField();
myTextField.y = 50;
addChild(myTextField);

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(-20,-20,40,40);
mySpriteCon.addChild(mySprite);
mySprite.x = 20;
mySprite.y = 20;
addChild(mySpriteCon);

mySprite.addEventListener(TransformGestureEvent.GESTURE_ROTATE, rothandler);

function rothandler(evt:TransformGestureEvent): void {
evt.target.parent.rotationZ += evt.target.rotation;
myTextField.text = evt.target.parent.rotation.toString();
}
```

Hinweis: *Nicht alle TransformGestureEvent-Eigenschaften werden in allen Laufzeitumgebungen unterstützt. Beispielsweise sind nicht alle berührungsempfindlichen Geräte in der Lage, eine Drehgeste auf dem Bildschirm zu erkennen. Deshalb wird die* `TransformGestureEvent.rotation`*-Eigenschaft auf diesen Geräten nicht unterstützt. Überprüfen Sie, ob bestimmte Eigenschaften unterstützt werden, um sicherzustellen, dass Ihre Anwendung funktioniert. Weitere Informationen finden Sie unter „Fehlerbehebung“ auf Seite 632.*

Gestenphasen

Gestenereignisse können auch über Phasen verfolgt werden; so können Sie während der Ausführung der Geste Eigenschaften verfolgen. Beispielsweise können Sie die x-Koordinaten verfolgen, während das Objekt mit einer Swipe-Bewegung verschoben wird. Verwenden Sie diese Werte, um nach Ende der Swipe-Bewegung eine Linie durch alle Punkte in der Bewegungsbahn zu zeichnen. Oder Sie können die visuelle Darstellung eines Anzeigeobjekts ändern, während es mit einer Schwenkgeste über den Bildschirm gezogen wird. Ändern Sie die Objektdarstellung erneut, wenn die Schwenkgeste abgeschlossen ist.

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;
var mySprite = new Sprite();
mySprite.addEventListener(TransformGestureEvent.GESTURE_PAN , onPan);
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0, 0, 40, 40);
var myTextField = new TextField();
myTextField.y = 200;
addChild(mySprite);
addChild(myTextField);

function onPan(evt:TransformGestureEvent):void {

    evt.target.localX++;

if (evt.phase==GesturePhase.BEGIN) {
    myTextField.text = "Begin";
    evt.target.scaleX *= 1.5;
    evt.target.scaleY *= 1.5;
}
if (evt.phase==GesturePhase.UPDATE) {
    myTextField.text = "Update";
    evt.target.alpha = 0.5;
}
if (evt.phase==GesturePhase.END) {
    myTextField.text = "End";
    evt.target.width = 40;
    evt.target.height = 40;
    evt.target.alpha = 1;
}
}
```

Hinweis: *Die Frequenz der Aktualisierungsphase richtet sich nach der Laufzeitumgebung. Bei einigen Kombinationen aus Betriebssystem und Hardware werden Aktualisierungen überhaupt nicht weitergegeben.*

Wert „all“ für Gestenphase bei einfachen Gestenereignissen

Bei einigen Gestenereignisobjekten werden einzelne Phasen des Gestenereignisses nicht verfolgt; stattdessen wird die phase-Eigenschaft des Ereignisobjekts mit dem Wert „all“ gefüllt. Bei einfachen Swipe-Gesten oder beim einfachen Tippen mit zwei Fingern wird das Ereignis nicht anhand von mehreren Phasen verfolgt. Die `phase`-Eigenschaft des Ereignisobjekts für ein InteractiveObject, das auf `gestureSwipe`- oder `gestureTwoFingerTap`-Ereignisse wartet, lautet immer `all`, nachdem das Ereignis ausgelöst wurde:

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;
var mySprite = new Sprite();
mySprite.addEventListener(TransformGestureEvent.GESTURE_SWIPE, onSwipe);
mySprite.addEventListener(GestureEvent.GESTURE_TWO_FINGER_TAP, onTwoTap);
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0, 0, 40, 40);
var myTextField = new TextField();
myTextField.y = 200;
addChild(mySprite);
addChild(myTextField);

function onSwipe(swipeEvt:TransformGestureEvent):void {
    myTextField.text = swipeEvt.phase // Output is "all"
}
function onTwoTap(tapEvt:GestureEvent):void {
    myTextField.text = tapEvt.phase // Output is "all"
}
```

Fehlerbehebung

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Die Hardware- und Software-Unterstützung für die Berührungseingabe ändert sich äußerst schnell. Diese Referenz enthält keine Liste aller Geräte und Kombinationen aus Betriebssystem und Software, die die Multitouch-Eingabe unterstützen. Sie enthält jedoch Anleitungen dazu, wie Sie mit der Discovery API feststellen können, ob Ihre Anwendung auf einem Gerät mit Multitouch-Unterstützung installiert ist. Weiterhin finden Sie Tipps zur Fehlerbehebung Ihres ActionScript-Codes.

Die Reaktion der Flash-Laufzeiten auf Berührungsereignisse richtet sich nach Informationen, die von dem Gerät, dem Betriebssystem oder der Software (wie einem Browser) an die Laufzeit übergeben werden. Wegen dieser Abhängigkeit von der Software-Umgebung ist es noch schwieriger, die Multitouch-Kompatibilität zu dokumentieren. Bestimmte Gesten oder Berührungen werden auf verschiedenen Geräten oft unterschiedlich interpretiert. Wird eine Drehung erkannt, wenn der Benutzer mit zwei Fingern gleichzeitig eine Drehbewegung ausführt? Oder wenn der Benutzer mit einem Finger einen Kreis auf dem Bildschirm zeichnet? Je nach der Hardware- und Software-Umgebung kann eine Drehgeste bei beiden Bewegungen – oder auch bei einer völlig anderen Bewegung – erkannt werden. Mit anderen Worten: Das Gerät teilt dem Betriebssystem die Art der Benutzereingabe mit und das Betriebssystem leitet diese Informationen an die Laufzeit weiter. Wenn die Laufzeit sich innerhalb eines Browsers befindet, wird das Gesten- oder Berührungsereignis in manchen Fällen von der Browsersoftware interpretiert, die die Eingabe nicht an die Laufzeit weitergibt. Dieses Verhalten ähnelt „Hotkeys“: Sie versuchen, in einem Browser über eine bestimmte Tastenkombination eine Aktion in Flash Player auszuführen, aber der Browser öffnet stattdessen ein Menü.

Bei einzelnen APIs und Klassen werden Sie darauf hingewiesen, wenn keine Kompatibilität mit bestimmten Betriebssystemen besteht. Sie können einzelne API-Einträge hier nachschlagen, wobei Sie mit der Multitouch-Klasse beginnen: http://help.adobe.com/de_DE/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html.

Im Folgenden werden einige häufige Gesten und Berührungen beschrieben:

Schwenken Bewegen Sie einen Finger von links nach rechts oder von rechts nach links. Auf einigen Geräten müssen Schwenkbewegungen mit zwei Fingern vorgenommen werden.

Drehen Berühren Sie das Display mit zwei Fingern, die Sie dann kreisförmig bewegen (zeichnen Sie also mit den Fingern einen imaginären Kreis auf dem Bildschirm nach). Der Drehpunkt befindet sich in der Mitte zwischen den beiden Punkten, die Sie mit den Fingern berühren.

Swipe Bewegen Sie drei Finger schnell von links nach rechts, von rechts nach links, von oben nach unten oder von unten nach oben.

Zoom Berühren Sie das Display mit zwei Fingern. Bewegen Sie die Finger dann auseinander, um die Anzeige zu vergrößern, oder aufeinander zu, um die Anzeige zu verkleinern.

Drücken und Tippen Drücken Sie mit einem Finger auf das Display und tippen Sie dann mit dem anderen Finger auf eine andere Stelle des Displays.

Jedes Gerät verfügt über eine eigene Dokumentation zu den unterstützten Gesten und zur Ausführung dieser Gesten. In den meisten Fällen muss der Benutzer zwischen zwei Gesten die Finger vom Display nehmen, um den Kontakt völlig zu unterbrechen (abhängig vom Betriebssystem).

Wenn Ihre Anwendung nicht auf Berührungsereignisse oder Gesten reagiert, überprüfen Sie Folgendes:

1. Haben Sie Ereignis-Listener für Berührungs- oder Gestenereignisse an eine Objektklasse angefügt, die von der InteractiveObject-Klasse erbt? Nur InteractiveObject-Instanzen können auf Berührungs- und Gestenereignisse warten.
2. Testen Sie Ihre Anwendung in Flash Professional CS5? Falls ja, versuchen Sie, die Anwendung zu veröffentlichen und zu testen, da Flash Professional die Interaktion abfangen kann.
3. Beginnen Sie mit einfachen Schritten und stellen Sie zunächst fest, was richtig funktioniert (das folgende Codebeispiel stammt aus dem API-Eintrag für `Multitouch.inputMode`:

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField()

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taplistener);

function taplistener(e:TouchEvent): void {
    myTextField.text = "I've been tapped";
    myTextField.y = 50;
    addChild(myTextField);
}
```

   Tippen Sie auf das Rechteck. Wenn dieses Beispiel funktioniert, wissen Sie, dass Ihre Umgebung ein einfaches Tippen unterstützt. Dann können Sie komplizierte Verarbeitungen testen.

   Das Testen der Gestenunterstützung ist komplizierter. Ein bestimmtes Gerät oder Betriebssystem unterstützt eine beliebige Kombination aus Gesteneingaben oder auch gar keine Gesteneingaben.

   Das folgende Beispiel zeigt einen einfachen Test für die Zoomgeste:

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;

stage.addEventListener(TransformGestureEvent.GESTURE_ZOOM , onZoom);
var myTextField = new TextField();
myTextField.y = 200;
myTextField.text = "Perform a zoom gesture";
addChild(myTextField);

function onZoom(evt:TransformGestureEvent):void {
    myTextField.text = "Zoom is supported";
}
```

Führen Sie eine Zoomgeste auf dem Gerät aus, um zu sehen, ob das Textfeld mit der Meldung `Zoom is supported` (Zoom wird unterstützt) ausgefüllt wird. Der Ereignis-Listener wird der Bühne hinzugefügt, sodass Sie die Geste in jedem Teil der Testanwendung ausführen können.

Das folgende Beispiel zeigt einen einfachen Test für die Schwenkgeste:

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;

stage.addEventListener(TransformGestureEvent.GESTURE_PAN , onPan);
var myTextField = new TextField();
myTextField.y = 200;
myTextField.text = "Perform a pan gesture";
addChild(myTextField);

function onPan(evt:TransformGestureEvent):void {
    myTextField.text = "Pan is supported";
}
```

Führen Sie eine Schwenkgeste auf dem Gerät aus, um zu sehen, ob das Textfeld mit der Meldung `Pan is supported` (Schwenken wird unterstützt) ausgefüllt wird. Der Ereignis-Listener wird der Bühne hinzugefügt, sodass Sie die Geste in jedem Teil der Testanwendung ausführen können.

Bei einigen Kombinationen aus Betriebssystem und Gerät werden beide Gesten, bei anderen nur eine Geste oder gar keine Geste unterstützt. Testen Sie dies in der Bereitstellungsumgebung Ihrer Anwendung.

Bekannte Probleme

Im Folgenden werden bekannte Probleme in Bezug auf die Berührungseingabe genannt:

1. Im mobilen Internet Explorer auf Windows Mobile-Betriebssystemen wird Inhalt in SWF-Dateien automatisch vergrößert oder verkleinert:

   Sie können dieses Zoomverhalten in Internet Explorer umgehen, indem Sie der HTML-Seite, die die SWF-Datei hostet, den folgenden Code hinzufügen:

   ```
   <head>
   <meta name="viewport" content="width=device-width, height=device-height, initial-
   scale=1.0">
   </head>
   ```

2. In Windows 7 (und möglicherweise auch in anderen Betriebssystemen) muss der Benutzer das Zeigegerät oder die Finger zwischen Gesten vom Bildschirm nehmen. Zum Drehen und Zoomen eines Bildes sind beispielsweise folgende Schritte erforderlich:
   - Führen Sie die Drehgeste durch.
   - Nehmen Sie Ihre Finger vom Bildschirm.
   - Berühren Sie den Bildschirm wieder mit den Fingern und führen Sie die Zoomgeste durch.
3. Unter Windows 7 (und möglicherweise auch anderen Betriebssystemen) generieren die Dreh- und Zoomgesten nicht immer eine Aktualisierungsphase, wenn die Geste sehr schnell durchgeführt wird.
4. Windows 7 Starter Edition bietet keine Multitouch-Unterstützung. Einzelheiten finden Sie im AIR Labs-Forum: http://forums.adobe.com/thread/579180?tstart=0
5. Auf Mac OS 10.5.3 und höher ist `Multitouch.supportsGestureEvents` immer `true`, auch wenn die Hardware keine Gestenereignisse unterstützt.

Kapitel 33: Kopieren und Einfügen

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Verwenden Sie die Klassen in der Zwischenablagen-API, um Informationen in die und aus der Zwischenablage des Systems zu kopieren. Zu den Datenformaten, die an eine oder aus einer in Adobe® Flash® Player oder Adobe® AIR® ausgeführten Anwendung übertragen werden können, zählen:

- Text
- HTML-formatierter Text
- Rich Text Format-Daten
- Serialisierte Objekte
- Objektverweise (nur gültig in der ursprünglichen Anwendung)
- Bitmaps (nur AIR)
- Dateien (nur AIR)
- URL-Strings (nur AIR)

Grundlagen zum Kopieren und Einfügen

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Die API für das Kopieren und Einfügen enthält die folgenden Klassen:

Paket	Klassen
flash.desktop	• Clipboard • ClipboardFormats • ClipboardTransferMode

Die statische Eigenschaft `Clipboard.generalClipboard` steht für die Zwischenablage des Betriebssystems. Die Clipboard-Klasse stellt Methoden für das Lesen und Schreiben von Daten in Clipboard-Objekten zur Verfügung.

Die HTMLLoader-Klasse (in AIR) und die TextField-Klasse implementieren Standardverhalten für die normalen Tastenkombinationen zum Kopieren und Einfügen. Um Verhaltensweisen bei Kopier- und Einfügeaktionen für benutzerdefinierte Komponenten anzuwenden, können Sie für diese Tasteneingaben einen Listener verwenden. Sie können auch native Menübefehle und Tastaturentsprechungen verwenden, um indirekt auf die Tastatureingaben zu reagieren.

Unterschiedliche Darstellungen derselben Informationen können in einem einzigen Clipboard-Objekt zur Verfügung gestellt werden, um anderen Anwendungen zu helfen, diese Daten zu verstehen und einzusetzen. Ein Bild kann zum Beispiel als Bilddaten, ein serialisiertes Bitmap-Objekt und als Datei aufgenommen werden. Die Darstellung der Daten in einem bestimmten Format kann zurückgestellt werden, sodass das Format erst dann erstellt wird, wenn die Daten in diesem Format gelesen werden.

Lesen aus der und Schreiben in die System-Zwischenablage

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Rufen Sie die `getData()`-Methode des `Clipboard.generalClipboard`-Objekts auf und übergeben Sie den Namen des zu lesenden Formats, um aus der Zwischenablage des Betriebssystems zu lesen:

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
    var text:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);
}
```

Hinweis: *Inhalte, die in Flash Player oder in einer anwendungsfremden Sandbox in AIR ausgeführt werden, können die `getData()`-Methode nur in einer Ereignisprozedur für ein `paste`-Ereignis aufrufen. Anders ausgedrückt, nur Code, der in der Sandbox der AIR-Anwendung ausgeführt wird, kann die `getData()`-Methode außerhalb einer `paste`-Ereignisprozedur aufrufen.*

Um Daten in die Zwischenablage zu schreiben, fügen Sie diese in einem oder mehreren Formaten in das Objekt `Clipboard.generalClipboard` ein. Bereits im selben Format bestehende Daten werden automatisch überschrieben. Es ist jedoch empfehlenswert, auch alle Daten in anderen Formaten aus der Zwischenablage des Systems zu entfernen, bevor Sie neue Daten in die Zwischenablage schreiben.

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

var textToCopy:String = "Copy to clipboard.";
Clipboard.generalClipboard.clear();
Clipboard.generalClipboard.setData(ClipboardFormats.TEXT_FORMAT, textToCopy, false);
```

Hinweis: *Inhalte, die in Flash Player oder in einer anwendungsfremden Sandbox in AIR ausgeführt werden, können die `setData()`-Methode nur in einer Ereignisprozedur für ein Benutzerereignis, wie etwa ein Tastatur- oder Mausereignis oder ein `copy`- oder `cut`-Ereignis aufrufen. Anders ausgedrückt, nur Code, der in der Sandbox der AIR-Anwendung ausgeführt wird, kann die `setData()`-Methode außerhalb einer Ereignisprozedur für ein Benutzerereignis aufrufen.*

Kopieren und Einfügen von HTML in AIR

Adobe AIR 1.0 und höher

Die HTML-Umgebung in Adobe AIR stellt eine eigene Reihe von Ereignissen und Standardverhalten für das Kopieren und Einfügen zur Verfügung. Nur Code, der in der Anwendungs-Sandbox ausgeführt wird, kann direkt über das AIR-Objekt `Clipboard.generalClipboard` auf die Zwischenablage des Systems zugreifen. JavaScript-Code in einer anwendungsfremden Sandbox kann über das Ereignisobjekt, das in Antwort auf eines der Kopier- oder Einfügeereignisse ausgelöst wird, das wiederum von einem Element in einem HTML-Dokument ausgelöst wurde, auf die Zwischenablage zugreifen.

Die Kopier- und Einfügeereignisse umfassen: `copy`, `cut` und `paste`. Das für diese Ereignisse ausgelöste Objekt bietet über die Eigenschaft `clipboardData` Zugriff auf die Zwischenablage.

Standardverhalten

Adobe AIR 1.0 und höher

AIR kopiert ausgewählte Einträge standardmäßig in Antwort auf einen Kopierbefehl, der durch eine Tastenkombination oder ein Kontextmenü ausgelöst werden kann. Innerhalb editierbarer Bereiche schneidet AIR bei einem Schneidebefehl Text aus oder fügt bei einem Einfügebefehl Text an der Zeigerposition oder Auswahl ein.

Um ein solches Standardverhalten zu verhindern, kann die Ereignisprozedur die Methode `preventDefault()` des ausgelösten Ereignisobjekts aufrufen.

Verwenden der clipboardData-Eigenschaft des Ereignisobjekts

Adobe AIR 1.0 und höher

Mit der Eigenschaft `clipboardData` des Ereignisobjekts, das infolge eines Kopier- oder Einfügeereignisses ausgelöst wurde, können Sie Daten auf der Zwischenablage lesen oder in diese schreiben.

Wenn Sie bei einem Kopier- oder Einfügeereignis Daten in die Zwischenablage schreiben wollen, verwenden Sie die Methode `setData()` des Objekts `clipboardData` und übergeben Sie die zu kopierenden Daten und den MIME-Typ:

```
function customCopy(event){
    event.clipboardData.setData("text/plain", "A copied string.");
}
```

Um auf Daten zuzugreifen, die eingefügt werden, verwenden Sie die Methode `getData()` des Objekts `clipboardData` und übergeben den MIME-Typ des Datenformats. Die zur Verfügung stehenden Formate werden durch die Eigenschaft `types` angegeben.

```
function customPaste(event){
    var pastedData = event.clipboardData("text/plain");
}
```

Auf die Methode `getData()` und die Eigenschaft `types` kann nur in Ereignisobjekten zugegriffen werden, die durch das Ereignis `paste` ausgelöst werden.

Im folgenden Beispiel wird dargestellt, wie das Standardverhalten beim Kopieren und Einfügen auf einer HTML-Seite außer Kraft gesetzt werden kann. Die Ereignisprozedur `copy` setzt den kopierten Text in Kursivschrift und kopiert ihn als HTML-Text in die Zwischenablage. Die Ereignisprozedur `cut` kopiert die ausgewählten Daten in die Zwischenablage und entfernt sie aus dem Dokument. Die Ereignisprozedur `paste` fügt den Inhalt der Zwischenablage als HTML ein und zeigt den eingefügten Text fett an.

```
 <html>
<head>
    <title>Copy and Paste</title>
    <script language="javascript" type="text/javascript">
        function onCopy(event){
            var selection = window.getSelection();
            event.clipboardData.setData("text/html","<i>" + selection + "</i>");
            event.preventDefault();
        }

        function onCut(event){
             var selection = window.getSelection();
             event.clipboardData.setData("text/html","<i>" + selection + "</i>");
             var range = selection.getRangeAt(0);
             range.extractContents();

            event.preventDefault();
        }

        function onPaste(event){
            var insertion = document.createElement("b");
            insertion.innerHTML = event.clipboardData.getData("text/html");
             var selection = window.getSelection();
             var range = selection.getRangeAt(0);
             range.insertNode(insertion);
            event.preventDefault();
        }
    </script>
</head>
<body onCopy="onCopy(event)"
     onPaste="onPaste(event)"
     onCut="onCut(event)">
<p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore
veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam
voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur
magni dolores eos qui ratione voluptatem sequi nesciunt.</p>
</body>
</html>
```

Datenformate in der Zwischenablage

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Mit den Zwischenablageformaten werden die Daten, die in ein Clipboard-Objekt platziert wurden, beschrieben. Die Standarddatenformate werden von Flash Player bzw. AIR automatisch zwischen ActionScript-Datentypen und Formaten für die Zwischenablage des Systems übersetzt. Außerdem können Anwendungsobjekte innerhalb und zwischen ActionScript-basierten Anwendungen mithilfe von anwendungsdefinierten Formaten übertragen werden.

Ein Clipboard-Objekt kann Darstellungen derselben Informationen in unterschiedlichen Formaten enthalten. Beispielswise könnte ein Clipboard-Objekt, dass ein „Sprite“ darstellt, Folgendes enthalten: ein Verweisformat zur Verwendung in derselben Anwendung, ein serialisiertes Format für eine andere in Flash Player oder AIR ausgeführte Anwendung, ein Bitmapformat für ein Bildbearbeitungsprogramm und ein Dateilistenformat, möglicherweise mit verzögerter Darstellung zur Kodierung einer PNG-Datei, um eine Darstellung des „Sprite“ in das Dateisystem kopieren oder ziehen zu können.

Standarddatenformate

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Die Konstanten für die Definition von Standardformatnamen werden durch die ClipboardFormats-Klasse bereitgestellt:

Konstante	Beschreibung
TEXT_FORMAT	Daten im Textformat werden in die und aus der String-Klasse von ActionScript übersetzt.
HTML_FORMAT	Text mit HTML-Code.
RICH_TEXT_FORMAT	Daten im Rich-Text-Format werden in die und aus der ByteArray-Klasse von ActionScript übersetzt. Der RTF-Code wird nicht interpretiert oder übersetzt.
BITMAP_FORMAT	(Nur AIR) Daten in diesem Format werden in die und aus der ActionScript-BitmapData-Klasse übersetzt.
FILE_LIST_FORMAT	(Nur AIR) Daten in diesem Format werden in die und aus der ActionScript-File-Klasse übersetzt.
URL_FORMAT	(Nur AIR) Daten in diesem Format werden in die und aus der ActionScript-String-Klasse übersetzt.

Wenn Daten in Reaktion auf ein `copy`-, `cut`- oder `paste`-Ereignis in HTML-Inhalt, der in einer AIR-Anwendung gehostet wird, kopiert und eingefügt werden, müssen anstelle der ClipboardFormat-Strings MIME-Typen verwendet werden. Folgende MIME-Typen sind für die Daten gültig:

MIME-Typ	Beschreibung
Text	"text/plain"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
File list	"application/x-vnd.adobe.air.file-list"

Hinweis: *Daten im Rich-Text-Format werden nicht über das Ereignis `clipboardData` des Ereignisobjekts, das infolge eines `paste`-Ereignisses in HTML-Inhalten ausgelöst wurde, zur Verfügung gestellt.*

Benutzerdefinierte Datenformate

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Sie können eigene anwendungsdefinierte Formate verwenden, um Objekte als Referenzen oder als serialisierte Kopien zu übertragen. Verweise sind nur innerhalb derselben Anwendung gültig. Serialisierte Objekte können zwischen Anwendungen übertragen werden, können jedoch nur mit Objekten verwendet werden, die gültig bleiben, wenn sie serialisiert und deserialisiert werden. Objekte können in der Regel serialisiert werden, wenn ihre Eigenschaften einfache Typen oder serialisierbare Objekte sind.

Um einem Clipboard-Objekt ein serialisiertes Objekt hinzuzufügen, stellen Sie den *serializable*-Parameter auf `true` ein, wenn Sie die `Clipboard.setData()`-Methode aufrufen. Der Formatname kann auf eines der Standardformate verweisen oder ein willkürlicher, von Ihrer Anwendung festgelegter String sein.

Übertragungsmodi

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Wenn ein Objekt in einem benutzerdefinierten Datenformat in die Zwischenablage geschrieben wird, können die Objektdaten als Verweis oder als serialisierte Kopie des Originalobjekts aus der Zwischenablage gelesen werden. Es gibt vier Übertragungsmodi, durch die festgelegt wird, ob Objekte als Verweise oder serialisierte Kopien übertragen werden:

Übertragungsmodus	Beschreibung
ClipboardTransferModes.ORIGINAL_ONLY	Es wird nur eine Referenz zurückgegeben. Stehen keine Referenzen zur Verfügung, wird ein Null-Wert zurückgegeben.
ClipboardTransferModes.ORIGINAL_PREFFERED	Wenn verfügbar, wird eine Referenz zurückgegeben. Wenn nicht, wird eine serialisierte Kopie zurückgegeben.
ClipboardTransferModes.CLONE_ONLY	Es wird nur eine serialisierte Kopie zurückgegeben. Steht keine serialisierte Kopie zur Verfügung, wird ein Null-Wert zurückgegeben.
ClipboardTransferModes.CLONE_PREFFERED	Wenn verfügbar, wird eine serialisierte Kopie zurückgegeben. Wenn nicht, wird eine Referenz zurückgegeben.

Lesen und Schreiben von benutzerdefinierten Datenformaten

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Wenn Sie ein Objekt in die Zwischenablage schreiben, können Sie einen beliebigen String, der nicht mit den reservierten Präfixen `air:` oder `flash:` beginnt, für den *format*-Parameter verwenden. Verwenden Sie denselben String wie für das Format, um das Objekt zu lesen Im folgenden Beispiel wird gezeigt, wie Objekte in die Zwischenablage geschrieben und aus dieser gelesen werden:

```
public function createClipboardObject(object:Object):Clipboard{
    var transfer:Clipboard = Clipboard.generalClipboard;
    transfer.setData("object", object, true);
}
```

Verwenden Sie denselben Formatnamen und den Übertragungsmodus `CLONE_ONLY` oder `CLONE_PREFFERED`, um ein serialisiertes Objekt (nach einem Zieh- oder Einfügevorgang) aus dem Clipboard-Objekt zu extrahieren.

```
var transfer:Object = clipboard.getData("object", ClipboardTransferMode.CLONE_ONLY);
```

Dem Clipboard-Objekt wird immer eine Referenz hinzugefügt. Um statt des serialisierten Objekts den Verweis aus dem Clipboard-Objekt zu extrahieren (nach einem Drop- oder Einfügevorgang), verwenden Sie den `ORIGINAL_ONLY`- oder `ORIGINAL_PREFFERED`-Übertragungsmodus.

```
var transferredObject:Object =
    clipboard.getData("object", ClipboardTransferMode.ORIGINAL_ONLY);
```

Verweise sind nur gültig, wenn das Clipboard-Objekt aus der aktuellen Anwendung stammt. Greifen Sie mit dem `ORIGINAL_PREFFERED`-Übertragungsmodus auf den Verweis zu, wenn er verfügbar ist, oder auf den serialisierten Klon, wenn der Verweis nicht verfügbar ist.

Zurückgestellte Wiedergabe

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Wenn die Erstellung eines Datenformats eine große Rechnerleistung erfordert, können Sie die zurückgestellte Wiedergabe verwenden, indem Sie eine Funktion angeben, die die Daten bei Bedarf zur Verfügung stellt. Diese Funktion wird nur dann aufgerufen, wenn der Empfänger eines Zieh- oder Einfügevorgangs die Daten im zurückgestellten Format anfordert.

Die Wiedergabefunktion wird mit der Methode `setDataHandler()` zum Clipboard-Objekt hinzugefügt. Die Funktion muss die Daten im entsprechenden Format zurückgeben. Haben Sie zum Beispiel `setDataHandler(ClipboardFormat.TEXT_FORMAT, writeText)` aufgerufen, dann muss die Funktion `writeText()` einen String zurückgeben.

Wird ein Datenformat desselben Typs mit der Methode `setData()` zum Clipboard-Objekt hinzugefügt, haben diese Daten Vorrang vor der zurückgestellten Version (die Wiedergabefunktion wird nie aufgerufen). Möglicherweise wird die Wiedergabefunktion aufgerufen, wenn auf dieselben Daten in der Zwischenablage ein zweites Mal zugegriffen wird.

Hinweis: *Unter Mac OS X kann die verzögerte Darstellung nur an benutzerdefinierten Datenformaten ausgeführt werden. Bei Standarddatenformaten wird die Darstellungsfunktion sofort aufgerufen.*

Einfügen von Text mit zurückgestellter Wiedergabefunktion

Flash Player 10 und höher, Adobe AIR 1.0 und höher

Im folgenden Beispiel wird gezeigt, wie die Funktion für die zurückgestellte Wiedergabe eingesetzt wird.

Beim Klicken auf die Schaltfläche „Kopieren“ wird der Inhalt der Zwischenablage des Systems gelöscht, um sicherzustellen, dass keine Daten von vorherigen Vorgängen in der Zwischenablage verbleiben. Anschließend wird mithilfe der `setDataHandler()`-Methode die `renderData()`-Funktion als Darstellungsfunktion der Zwischenablage eingestellt.

Wenn der Benutzer im Kontextmenü des Zieltextfelds den Befehl „Einfügen“ auswählt, greift die Anwendung auf die Zwischenablage zu und legt den Zieltext fest. Da das Textdatenformat in der Zwischenablage nicht mit einem String sondern einer Funktion eingestellt wurde, ruft die Zwischenablage die `renderData()`-Funktion auf. Die Funktion `renderData()` gibt den Text im Quelltext zurück, der dann einem Zieltext zugeordnet wird.

Beachten Sie, das Bearbeitungsvorgänge am Quelltext vor dem Betätigen der Schaltfläche „Einfügen“ im eingefügten Text sichtbar sind, auch wenn die Änderungen nach dem Betätigen der Schaltfläche „Kopieren“ getätigt wurden. Die Wiedergabefunktion kopiert den Quelltext erst, wenn die Schaltfläche „Einfügen“ betätigt wird. (Bei Verwendung der zurückgestellten Wiedergabe in einer echten Anwendung sollten Sie daher die Quelldaten möglicherweise speichern oder schützen, um dieses Problem zu vermeiden.)

Flash-Beispiel

```
package {
   import flash.desktop.Clipboard;
   import flash.desktop.ClipboardFormats;
   import flash.desktop.ClipboardTransferMode;
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.text.TextFormat;
   import flash.text.TextFieldType;
   import flash.events.MouseEvent;
   import flash.events.Event;
   public class DeferredRenderingExample extends Sprite
   {
       private var sourceTextField:TextField;
       private var destination:TextField;
       private var copyText:TextField;
       public function DeferredRenderingExample():void
       {
           sourceTextField = createTextField(10, 10, 380, 90);
           sourceTextField.text = "Neque porro quisquam est qui dolorem "
               + "ipsum quia dolor sit amet, consectetur, adipisci velit.";

           copyText = createTextField(10, 110, 35, 20);
           copyText.htmlText = "<a href='#'>Copy</a>";
           copyText.addEventListener(MouseEvent.CLICK, onCopy);

           destination = createTextField(10, 145, 380, 90);
           destination.addEventListener(Event.PASTE, onPaste);
       }
       private function createTextField(x:Number, y:Number, width:Number,
                       height:Number):TextField
       {
           var newTxt:TextField = new TextField();
           newTxt.x = x;
           newTxt.y = y;
           newTxt.height = height;
           newTxt.width = width;
           newTxt.border = true;
           newTxt.multiline = true;
           newTxt.wordWrap = true;
           newTxt.type = TextFieldType.INPUT;
           addChild(newTxt);
           return newTxt;
       }
       public function onCopy(event:MouseEvent):void
       {
           Clipboard.generalClipboard.clear();
           Clipboard.generalClipboard.setDataHandler(ClipboardFormats.TEXT_FORMAT,
                           renderData);
       }
       public function onPaste(event:Event):void
       {
```

```
            sourceTextField.text =
            Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT).toString;
        }
        public function renderData():String
        {
            trace("Rendering data");
            var sourceStr:String = sourceTextField.text;
            if (sourceTextField.selectionEndIndex >
                     sourceTextField.selectionBeginIndex)
            {
                return sourceStr.substring(sourceTextField.selectionBeginIndex,
                                    sourceTextField.selectionEndIndex);
            }
            else
            {
                return sourceStr;
            }
        }
    }
}
```

Flex-Beispiel

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" width="326"
height="330" applicationComplete="init()">
    <mx:Script>
    <![CDATA[
    import flash.desktop.Clipboard;
    import flash.desktop.ClipboardFormats;

    public function init():void
    {
        destination.addEventListener("paste", doPaste);
    }

    public function doCopy():void
    {
        Clipboard.generalClipboard.clear();
        Clipboard.generalClipboard.setDataHandler(ClipboardFormats.TEXT_FORMAT, renderData);
    }
    public function doPaste(event:Event):void
    {
        destination.text =
Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT).toString;
    }

    public function renderData():String{
        trace("Rendering data");
        return source.text;
    }
    ]]>
    </mx:Script>
    <mx:Label x="10" y="10" text="Source"/>
    <mx:TextArea id="source" x="10" y="36" width="300" height="100">
        <mx:text>Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur,
adipisci velit.</mx:text>
    </mx:TextArea>
    <mx:Label x="10" y="181" text="Destination"/>
    <mx:TextArea id="destination"  x="12" y="207" width="300" height="100"/>
    <mx:Button click="doCopy();" x="91" y="156" label="Copy"/>
</mx:Application>
```

Kapitel 34: Accelerometer-Eingabe

Flash Player 10.1 und höher, Adobe AIR 2 und höher

Die Accelerometer-Klasse setzt Ereignisse ab, die auf einer vom Bewegungssensor des Geräts erkannten Aktivität basieren. Diese Daten repräsentieren die Position des Geräts oder die Bewegung entlang einer dreidimensionalen Achse. Wenn das Gerät bewegt wird, erkennt der Sensor diese Bewegung und gibt die Beschleunigungskoordination des Geräts zurück. Die Accelerometer-Klasse bietet Methoden zum Abfragen, ob ein Beschleunigungsmesser unterstützt wird, und zum Festlegen der Rate, mit der Beschleunigungsereignisse ausgelöst werden.

Die Achsen des Beschleunigungsmessers werden in Bezug auf die Ausrichtung der Anzeige normalisiert, nicht in Bezug auf die Ausrichtung des Geräts. Wenn die Anzeige auf dem Gerät neu ausgerichtet wird, werden auch die Achsen des Beschleunigungsmessers neu ausgerichtet. Deshalb verläuft die y-Achse immer mehr oder weniger vertikal, wenn der Benutzer das Gerät in einer normalen, aufrechten Anzeigeposition hält, unabhängig davon, in welche Richtung das Telefon gedreht wird. Wenn die automatische Ausrichtung deaktiviert ist (beispielsweise wenn der SWF-Inhalt im Browser im Vollbildmodus angezeigt wird), werden die Achsen des Beschleunigungsmessers beim Drehen des Geräts nicht neu ausgerichtet.

Verwandte Hilfethemen

flash.sensors.Accelerometer

flash.events.AccelerometerEvent

Überprüfen der Accelerometer-Unterstützung

Mithilfe der `Accelerometer.isSupported`-Eigenschaft können Sie überprüfen, ob die Laufzeitumgebung dieses Funktionsmerkmal unterstützt:

```
if (Accelerometer.isSupported)
{
    // Set up Accelerometer event listeners and code.
}
```

Die Accelerometer-Klasse und ihre Mitglieder stehen in den Laufzeitversionen zur Verfügung, die für die jeweiligen API-Einträge aufgelistet sind. Die jeweils aktuelle Umgebung zur Laufzeit bestimmt jedoch, ob dieses Funktionsmerkmal verfügbar ist. So können Sie beispielsweise Code mit den Eigenschaften der Accelerometer-Klasse für Flash Player 10.1 kompilieren, aber Sie müssen mit der `Accelerometer.isSupported`-Eigenschaft überprüfen, ob das Accelerometer-Funktionsmerkmal auf dem Endgerät zur Verfügung steht. Wenn `Accelerometer.isSupported` zur Laufzeit den Wert `true` hat, wird Accelerometer zurzeit unterstützt.

Erkennen von Accelerometer-Änderungen

Zur Verwendung des Sensors für den Beschleunigungsmesser instanziieren Sie ein Accelerometer-Objekt und registrieren Sie die von diesem Objekt ausgelösten `update`-Ereignisse. Das `update`-Ereignis ist ein Accelerometer-Ereignisobjekt. Das Ereignis hat vier Eigenschaften, bei denen es sich jeweils um Zahlen handelt:

- `accelerationX` – Beschleunigung entlang der x-Achse, gemessen in G. Die x-Achse verläuft von der linken zur rechten Seite des Geräts, wenn es sich in der aufrechten Position befindet. (Das Gerät befindet sich in der aufrechten Position, wenn die Oberseite des Geräts nach oben zeigt.) Die Beschleunigung ist positiv, wenn das Gerät sich nach rechts bewegt.
- `accelerationY` – Beschleunigung entlang der y-Achse, gemessen in G. Die y-Achse verläuft von der unteren zur oberen Seite des Gerät, wenn es sich in der aufrechten Position befindet. (Das Gerät befindet sich in der aufrechten Position, wenn die Oberseite des Geräts nach oben zeigt.) Die Beschleunigung ist positiv, wenn sich das Gerät in Relation zu dieser Achse nach oben bewegt.
- `accelerationZ` – Beschleunigung entlang der z-Achse, gemessen in G. Die z-Achse verläuft senkrecht zur Oberfläche des Geräts. Die Beschleunigung ist positiv, wenn Sie das Gerät so halten, dass seine Oberseite nach oben zeigt. Die Beschleunigung ist negativ, wenn die Oberseite des Geräts zum Boden zeigt.
- `timestamp` – Die Anzahl der Millisekunden seit der Initialisierung der Laufzeitumgebung bis zum Auftreten des Ereignisses.

1 g ist die Standardbeschleunigung aufgrund der Schwerkraft, ungefähr 9,8 m/s^2.

Im Folgenden sehen Sie ein einfaches Beispiel zur Anzeige von Daten des Beschleunigungsmessers in einem Textfeld:

```
var accl:Accelerometer;
if (Accelerometer.isSupported)
{
    accl = new Accelerometer();
    accl.addEventListener(AccelerometerEvent.UPDATE, updateHandler);
}
else
{
    accTextField.text = "Accelerometer feature not supported";
}
function updateHandler(evt:AccelerometerEvent):void
{
    accTextField.text = "acceleration X: " + evt.accelerationX.toString() + "\n"
            + "acceleration Y: " + evt.accelerationY.toString() + "\n"
            + "acceleration Z: " + evt.accelerationZ.toString()
}
```

Bevor Sie diesen Code verwenden können, müssen Sie das `accTextField`-Textfeld erstellen und der Anzeigeliste hinzufügen.

Sie können das gewünschte Zeitintervall für Ereignisse des Beschleunigungsmessers anpassen, indem Sie die `setRequestedUpdateInterval()`-Methode für das Accelerometer-Objekt aufrufen. Diese Methode akzeptiert einen Parameter, `interval`. Dies ist das angeforderte Aktualisierungsintervall in Millisekunden:

```
var accl:Accelerometer;
accl = new Accelerometer();
accl.setRequestedUpdateInterval(1000);
```

Die tatsächliche Zeitspanne zwischen Aktualisierungen des Beschleunigungsmessers kann größer oder kleiner als dieser Wert sein. Änderungen am Aktualisierungsintervall betreffen alle registrierten Listener. Wenn Sie die `setRequestedUpdateInterval()`-Methode nicht aufrufen, werden die Aktualisierungen in der Anwendung gemäß dem Standardintervall des Geräts durchgeführt.

Die Accelerometer-Daten sind nicht hundertprozentig genau. Sie können einen gleitenden Mittelwert der neuesten Daten verwenden, um Ungenauigkeiten auszugleichen. Im folgenden Beispiel werden beispielsweise kürzlich erhaltene Daten des Beschleunigungsmessers bei den aktuellen Daten berücksichtigt, um ein gerundetes Ergebnis zu erhalten:

```
var accl:Accelerometer;
var rollingX:Number = 0;
var rollingY:Number = 0;
var rollingZ:Number = 0;
const FACTOR:Number = 0.25;

if (Accelerometer.isSupported)
{
    accl = new Accelerometer();
    accl.setRequestedUpdateInterval(200);
    accl.addEventListener(AccelerometerEvent.UPDATE, updateHandler);
}
else
{
    accTextField.text = "Accelerometer feature not supported";
}
function updateHandler(event:AccelerometerEvent):void
{
    accelRollingAvg(event);
    accTextField.text = rollingX + "\n" +  rollingY + "\n" + rollingZ + "\n";
}

function accelRollingAvg(event:AccelerometerEvent):void
{
    rollingX = (event.accelerationX * FACTOR) + (rollingX * (1 - FACTOR));
    rollingY = (event.accelerationY * FACTOR) + (rollingY * (1 - FACTOR));
    rollingZ = (event.accelerationZ * FACTOR) + (rollingZ * (1 - FACTOR));
}
```

Ein solcher gleitender Mittelwert wird jedoch nur bei einem kurzen Aktualisierungsintervall des Beschleunigungsmessers empfohlen.

Kapitel 35: Ziehen und Ablegen in AIR

Adobe AIR 1.0 und höher

Verwenden Sie die Klassen in der Adobe® AIR™ Drag-and-Drop-API, um Drag & Drop-Bewegungen in der Benutzeroberfläche zu unterstützen. Eine *Bewegung* (auch „Geste“) bezeichnet in diesem Sinn eine Benutzeraktion, die sowohl über das Betriebssystem als auch über die AIR-Anwendung herbeigeführt wird und die Absicht des Benutzers ausdrückt, Informationen zu kopieren, zu verschieben oder zu verknüpfen. Die Bewegung *Herausziehen* wird ausgeführt, wenn der Benutzer ein Objekt aus einer Komponente oder Anwendung herauszieht. Die Bewegung *Hineinziehen* wird ausgeführt, wenn der Benutzer ein Objekt von außerhalb in eine Komponente oder Anwendung hineinzieht.

Mit der Drag & Drop-API können Sie den Benutzern die Möglichkeit geben, Daten zwischen Anwendungen und zwischen den Komponenten einer Anwendung im Rahmen bestimmter Aufgaben und Funktionen durch Ziehen auszutauschen. Dabei werden folgende Übertragungsformate unterstützt:

- Bitmaps
- Dateien
- HTML-formatierter Text
- Text
- Rich Text Format-Daten
- URLs
- Dateizusage
- Serialisierte Objekte
- Objektverweise (nur gültig in der ursprünglichen Anwendung)

Grundlagen zur Drag & Drop-Funktion in AIR

Adobe AIR 1.0 und höher

Eine kurze Erklärung sowie Codebeispiele zum Drag & Drop in AIR-Anwendungen finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Supporting drag-and-drop and copy-and-paste (Flex)
- Supporting drag-and-drop and copy-and-paste (Flash)

Die Drag & Drop-API umfasst die folgenden Klassen:

Paket	Klassen
flash.desktop	• NativeDragManager • NativeDragOptions • Clipboard • URLFilePromise • IFilePromise Innerhalb der Drag & Drop-API verwendete Konstanten werden in den folgenden Klassen definiert: • NativeDragActions • ClipboardFormat • ClipboardTransferModes
flash.events	NativeDragEvent

Phasen der Drag & Drop-Bewegungen

Die Drag & Drop-Bewegung setzt sich aus drei Phasen zusammen:

Initiation *Der Benutzer initiiert eine Drag & Drop-Operation, indem er von einer Komponente oder einem Element in einer Komponente aus zu Ziehen beginnt und dabei die Maustaste gedrückt hält.* Die Komponente, aus der das gezogene Element stammt, wird in der Regel mit „Ziehinitiator“ bezeichnet, denn sie löst die Ereignisse `nativeDragStart` und `nativeDragComplete` aus. Eine Adobe AIR-Anwendung beginnt eine Ziehoperation, indem sie die Methode `NativeDragManager.doDrag()` als Reaktion auf ein `mouseDown`- oder `mouseMove`-Ereignis aufruft.

Wenn die Ziehoperation außerhalb einer AIR-Anwendung eingeleitet wurde, gibt es kein Initiatorobjekt, das `nativeDragStart`- oder `nativeDragComplete`-Ereignisse auslösen kann.

Ziehen *Während die Maustaste noch gehalten wird, verschiebt der Benutzer den Mauscursor auf eine andere Komponente, Anwendung oder den Desktop.* Solange das Element gezogen wird, löst das Initiatorobjekt kontinuierlich `nativeDragUpdate`-Ereignisse aus. (Dieses Ereignis wird in AIR für Linux nicht ausgelöst.) Wenn der Benutzer den Mauscursor über ein mögliches Ablageziel in einer AIR-Anwendung führt, löst das Ablageziel ein `nativeDragEnter`-Ereignis aus. Die Ereignisprozedur kann das Ereignisobjekt untersuchen, um festzustellen, ob die gezogenen Daten in einem Format vorliegen, das vom Ziel akzeptiert wird. Ist dies der Fall, kann man dem Benutzer durch Aufruf der Methode `NativeDragManager.acceptDragDrop()` die Möglichkeit zum Ablegen der Daten geben.

Solange die Ziehbewegung über einem interaktiven Objekt bleibt, löst dieses kontinuierlich `nativeDragOver`-Ereignisse aus. Verlässt die Ziehbewegung das interaktive Objekt, löst es ein `nativeDragExit`-Ereignis aus.

Ablegen *Der Benutzer lässt die Maustaste über dem gewünschten Ablageziel los.* Handelt es sich bei dem Ziel um eine AIR-Anwendung oder -Komponente, so löst das Zielobjekt ein `nativeDragDrop`-Ereignis aus. Die Ereignisprozedur kann auf die übertragenen Daten aus dem Ereignisobjekt zugreifen. Befindet sich das Ziel außerhalb von AIR, wird die Drop-Operation vom Betriebssystem oder von einer anderen Anwendung abgewickelt. In beiden Fällen löst das initiierende Objekt ein `nativeDragComplete`-Ereignis aus (wenn die Ziehoperation aus AIR heraus gestartet wurde).

Die NativeDragManager-Klasse steuert sowohl das Hereinziehen als auch das Herausziehen. Alle Mitglieder der NativeDragManager-Klasse sind statisch; erstellen Sie keine Instanz dieser Klasse.

Das Clipboard-Objekt

Daten, die in eine Anwendung oder Komponente hineingezogen oder aus einer Anwendung oder Komponente herausgezogen werden, befinden sich in einem Clipboard-Objekt. Ein einzelnes Clipboard-Objekt kann mehrere Repräsentationen derselben Information verfügbar machen, sodass sich die Wahrscheinlichkeit, dass eine andere Anwendung diese Daten interpretieren und weiterverarbeiten kann, erhöht wird. Ein Bild beispielsweise könnte in Form von Bilddaten, als serialisiertes Bitmap-Objekt oder als Datei eingefügt werden. Das Rendern der Daten in einem bestimmten Format kann an eine Renderfunktion weitergereicht werden, die aber erst aufgerufen wird, wenn die Daten gelesen werden.

Nachdem eine Ziehbewegung gestartet wurde, kann nur eine Ereignisprozedur für die Ereignisse `nativeDragEnter`, `nativeDragOver` oder `nativeDragDrop` auf das Clipboard-Objekt zugreifen. Nachdem die Ziehbewegung beendet wird, kann das Clipboard-Objekt nicht mehr gelesen oder wiederverwendet werden.

Ein Anwendungsobjekt kann als Verweis übertragen werden oder als serialisiertes Objekt. Verweise sind nur innerhalb der Ursprungsanwendung zulässig. Serialisierte Objektübertragungen zwischen AIR-Anwendungen sind zulässig, können aber nur bei Objekten verwendet werden, die gültig bleiben, wenn sie serialisiert und deserialisiert werden. Objekte, die serialisiert werden, werden in das Action Message Format für ActionScript 3 (AMF3) konvertiert, einem stringbasierten Datenübertragungsformat.

Arbeiten mit der Flex-Architektur

Beim Erstellen von Flex-Anwendungen ist es meist besser, die Adobe® Flex™-Drag & Drop-API zu verwenden. Die Flex-Architekur stellt ein entsprechendes Set an Leistungsmerkmalen zur Verfügung, wenn in AIR eine Flex-Anwendung ausgeführt wird (sie verwendet den AIR-NativeDragManager intern). Flex unterhält ein begrenzteres Set an Leistungsmerkmalen, wenn eine Anwendung oder Komponente innerhalb der restriktiveren Browserumgebung ausgeführt wird. AIR-Klassen können nicht in Komponenten oder Anwendungen verwendet werden, die außerhalb der AIR-Laufzeitumgebung verwendet werden.

Unterstützung des Herausziehens

Adobe AIR 1.0 und höher

Damit das „Herausziehen“ unterstützt wird, müssen Sie als Reaktion auf ein `mouseDown`-Ereignis ein Clipboard-Objekt erstellen und es an die Methode `NativeDragManager.doDrag()` senden. Die Anwendung kann dann auf das `nativeDragComplete`-Ereignis des initialisierenden Objekts warten, um zu bestimmen, welche Aktion auszuführen ist, wenn der Benutzer die Bewegung abschließt oder auch abbricht.

Vorbereiten der Daten für die Übertragung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um Daten oder ein Objekt für eine Ziehoperation vorzubereiten, erstellen Sie ein Clipboard-Objekt und fügen die zu übertragenden Informationen in einem oder mehreren Formaten hinzu. Sie können die Standarddatenformate verwenden, um Daten weiterzuleiten, die dann automatisch in native Clipboard-Formate oder von der Anwendung definierte Formate übersetzt werden können.

Wenn die Konvertierung der zu übertragenden Informationen in ein bestimmtes Format zu rechenintensiv ist, können Sie den Namen einer Prozedurfunktion angeben, die die Konvertierung ausführt. Die Funktion wird nur dann aufgerufen, wenn das verbundene Format von der empfangenden Komponente oder Anwendung gelesen wird.

Weitere Informationen zu Formaten, die in der Zwischenablage verwendet werden können, finden Sie unter „Datenformate in der Zwischenablage“ auf Seite 638.

Das folgende Beispiel illustriert die Erstellung eines Clipboard-Objekts, das eine Bitmap in mehreren Formaten enthält: ein Bitmap-Objekt, ein natives Bitmap-Format und ein Dateilistenformat mit der Datei, aus der die Bitmap ursprünglich geladen wurde:

```
import flash.desktop.Clipboard;
import flash.display.Bitmap;
import flash.filesystem.File;
public function createClipboard(image:Bitmap, sourceFile:File):Clipboard{
    var transfer:Clipboard = new Clipboard();
    transfer.setData("CUSTOM_BITMAP", image, true); //Flash object by value and by reference
    transfer.setData(ClipboardFormats.BITMAP_FORMAT, image.bitmapData, false);
    transfer.setData(ClipboardFormats.FILE_LIST_FORMAT, new Array(sourceFile), false);
    return transfer;
}
```

Starten einer Herausziehoperation

Adobe AIR 1.0 und höher

Um eine Ziehoperation zu starten, rufen Sie als Reaktion auf ein mouseDown-Ereignis die Methode `NativeDragManager.doDrag()` auf. Die `doDrag()`-Methode ist eine statische Methode, die die folgenden Parameter akzeptiert:

Parameter	Beschreibung
initiator	Das Objekt, bei dem die Ziehbewegung anfing, und das die Ereignisse `dragStart` und `dragComplete` auslöst. Der Initiator muss ein interaktives Objekt sein.
clipboard	Das Clipboard-Objekt mit den zu übertragenden Daten. In den NativeDragEvent-Objekten, die während der Drag & Drop-Sequenz ausgelöst wurden, wird auf das Clipboard-Objekt verwiesen.
dragImage	(Optional) Ein BitmapData-Objekt, das beim Ziehen angezeigt wird. Das Bild kann einen `alpha`-Wert angeben. (Hinweis: Unter Microsoft Windows wird den Ziehbildern immer ein fester Alphawert für die Abblendung zugewiesen).
offset	(Optional) Ein Point-Objekt, das den Versatz des Ziehbilds vom Maus-Hotspot angibt. Verwenden Sie negative Koordinaten, um das Ziehbild in Relation zum Mauscursor nach oben und nach links zu versetzen. Wenn kein Versatz angegeben ist, wird die obere linke Ecke des Ziehbilds am Maus-Hotspot positioniert.
actionsAllowed	(Optional) Ein NativeDragOptions-Objekt, das festlegt, welche Aktionen (Kopieren, Verschieben oder Verknüpfen) für die Ziehoperation zulässig sind. Wenn kein Argument angegeben wird, sind alle Aktionen erlaubt. In den NativeDragEvent-Objekten wird auf das DragOptions-Objekt verwiesen, damit ein potentielles Ablageziel überprüfen kann, ob die zulässigen Aktionen mit dem Zweck der Zielkomponente kompatibel sind. Eine Papierkorbkomponente etwa wird wohl nur Ziehbewegungen akzeptieren, die ein Verschieben erlauben.

Das folgende Beispiel zeigt, wie Sie eine Ziehoperation für ein aus einer Datei geladenes Bitmap-Objekt starten: Das Beispiel lädt ein Bild und startet die Ziehoperation aufgrund eines `mouseDown`-Ereignisses.

```
 package
{
import flash.desktop.NativeDragManager;
import mx.core.UIComponent;
import flash.display.Sprite;
import flash.display.Loader;
import flash.system.LoaderContext;
import flash.net.URLRequest;
import flash.geom.Point;
import flash.desktop.Clipboard;
import flash.display.Bitmap;
import flash.filesystem.File;
import flash.events.Event;
import flash.events.MouseEvent;

public class DragOutExample extends UIComponent Sprite {
    protected var fileURL:String = "app:/image.jpg";
    protected var display:Bitmap;

    private function init():void {
        loadImage();
    }
    private function onMouseDown(event:MouseEvent):void {
        var bitmapFile:File = new File(fileURL);
        var transferObject:Clipboard = createClipboard(display, bitmapFile);
        NativeDragManager.doDrag(this,
                            transferObject,
                            display.bitmapData,
                            new Point(-mouseX,-mouseY));
    }
    public function createClipboard(image:Bitmap, sourceFile:File):Clipboard {
        var transfer:Clipboard = new Clipboard();
        transfer.setData("bitmap",
                            image,
                            true);
                            // ActionScript 3 Bitmap object by value and by reference
        transfer.setData(ClipboardFormats.BITMAP_FORMAT,
                            image.bitmapData,
                            false);
                            // Standard BitmapData format
        transfer.setData(ClipboardFormats.FILE_LIST_FORMAT,
```

```
                            new Array(sourceFile),
                            false);
                            // Standard file list format
        return transfer;
    }
    private function loadImage():void {
        var url:URLRequest = new URLRequest(fileURL);
        var loader:Loader = new Loader();
        loader.load(url,new LoaderContext());
        loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoadComplete);
    }
    private function onLoadComplete(event:Event):void {
        display = event.target.loader.content;
        var flexWrapper:UIComponent = new UIComponent();
        flexWrapper.addChild(event.target.loader.content);
        addChild(flexWrapper);
        flexWrapper.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
    }
}
}
```

Abschließen einer Herausziehübertragung

Adobe AIR 1.0 und höher

Wenn ein Benutzer das gezogene Element ablegt, indem er die Maustaste loslässt, löst das Initiatorobjekt ein `nativeDragComplete`-Ereignis aus. Sie können die `dropAction`-Eigenschaft des Ereignisobjekts überprüfen und dann die entsprechende Aktion durchführen. Wenn die Aktion zum Beispiel `NativeDragAction.MOVE` ist, könnten Sie die Quelle von ihrer ursprünglichen Position entfernen. Der Benutzer kann eine Ziehbewegung abbrechen, indem er die Maustaste loslässt, solange sich der Cursor außerhalb eines zulässigen Ablageziels befindet. Der Ziehmanager setzt die `dropAction`-Eigenschaft einer abgebrochenen Bewegung auf `NativeDragAction.NONE`.

Unterstützung des Hineinziehens

Adobe AIR 1.0 und höher

Damit das „Hineinziehen“ unterstützt wird, muss Ihre Anwendung (bzw. typischerweise eine visuelle Komponente der Anwendung) auf die Ereignisse `nativeDragEnter` oder `nativeDragOver` reagieren.

Die einzelnen Schritte in einer typischen Ablegoperation

Adobe AIR 1.0 und höher

Ein Ablegoperation umfasst typischerweise folgende Ereignissequenz:

1. Der Benutzer zieht ein Clipboard-Objekt über eine Komponente.
2. Die Komponente löst ein `nativeDragEnter`-Ereignis aus.
3. Die Ereignisprozedur `nativeDragEnter` untersucht das Ereignisobjekt, um die verfügbaren Datenformate und die zulässigen Aktionen zu überprüfen. Ist die Komponente in der Lage, das abzulegende Objekt weiterzuverarbeiten, ruft sie die Methode `NativeDragManager.acceptDragDrop()` auf.

4 Der NativeDragManager ändert den Mauscursor, um anzuzeigen, dass das Objekt abgelegt werden kann.
5 Der Benutzer legt das Objekt über eine Komponente ab.
6 Die empfangende Komponente löst ein `nativeDragDrop`-Ereignis aus.
7 Die empfangende Komponente liest die Daten im gewünschten Format aus dem Clipboard-Objekt innerhalb des Ereignisobjekts.
8 Hat die Ziehbewegung ihren Ursprung in einer AIR-Anwendung, löst das initiierende interaktive Objekt ein `nativeDragComplete`-Ereignis aus. Hat die Ziehbewegung ihren Ursprung außerhalb von AIR, wird keine Rückmeldung gesendet.

Bestätigen des Hineinziehens

Adobe AIR 1.0 und höher

Wenn ein Benutzer ein Zwischenablageelement in eine visuelle Komponente hineinzieht, löst die Komponente `nativeDragEnter`- und `nativeDragOver`-Ereignisse aus. Um zu bestimmen, ob die Komponente das Zwischenablageelement akzeptieren kann, können die Prozeduren für diese Ereignisse die Eigenschaften `clipboard` und `allowedActions` dieses Ereignisobjekts überprüfen. Um zu signalisieren, dass die Komponente das abzulegende Element akzeptieren kann, muss die Ereignisprozedur die Methode `NativeDragManager.acceptDragDrop()` aufrufen und einen Verweis an die empfangende Komponente übergeben. Wenn mehrere registrierte Ereignis-Listener die `acceptDragDrop()`-Methode aufrufen, hat die letzte Prozedur in der Liste den Vorrang. Der `acceptDragDrop()`-Aufruf bleibt gültig, bis der Mauscursor die Grenzen des empfangenden Objekts verlässt und dabei das `nativeDragExit`-Ereignis auslöst.

Wenn im `allowedActions`-Parameter, der an `doDrag()` übergeben wurde, mehrere Aktionen zugelassen sind, kann der Benutzer auswählen, welche der zulässigen Aktionen durchgeführt werden soll, indem er eine Zusatztaste drückt. Der Ziehmanager ändert das Cursorbild, um dem Benutzer mitzuteilen, welche Aktion ausgeführt wird, wenn er die Ziehoperation abschließt. Die beabsichtigte Aktion wird von der `dropAction`-Eigenschaft des NativeDragEvent-Objekts gemeldet. Die für eine Ziehbewegung eingestellte Aktion ist nur ein Vorschlag. Die an der Übertragung beteiligten Komponenten müssen das entsprechende Verhalten implementieren. Um beispielsweise eine Verschiebaktion abzuschließen, würde die Ursprungsanwendung (der Ziehinitiator) das gezogene Element wohl entfernen und das Ablageziel würde es hinzufügen.

Die Zielanwendung kann die Ablegaktion auf drei mögliche Aktionen begrenzen, indem sie die `dropAction`-Eigenschaft der NativeDragManager-Klasse setzt. Wenn ein Benutzer versucht, über die Tastatur eine andere Aktion zu wählen, zeigt der NativeDragManager den *Nicht-verfügbar*-Cursor an. Setzen Sie die Eigenschaft `dropAction` in den Prozeduren, und zwar sowohl für das Ereignis `nativeDragEnter` als auch für das Ereignis `nativeDragOver`.

Das folgende Beispiel präsentiert eine Ereignisprozedur für ein `nativeDragEnter`- oder `nativeDragOver`-Ereignis. Diese Prozedur akzeptiert eine Hineinziehbewegung, wenn die gezogene Zwischenablage Daten im Textformat enthält.

```
import flash.desktop.NativeDragManager;
import flash.events.NativeDragEvent;

public function onDragIn(event:NativeDragEvent):void{
    NativeDragManager.dropAction = NativeDragActions.MOVE;
    if(event.clipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
        NativeDragManager.acceptDragDrop(this); //'this' is the receiving component
    }
}
```

Abschließen der Ablegaktion

Adobe AIR 1.0 und höher

Wenn ein Benutzer das gezogene Element auf dem interaktiven Objekt ablegt, das die Ziehbewegung akzeptiert hat, löst das interaktive Objekt ein `nativeDragDrop`-Ereignis aus. Die Ereignisprozedur für dieses Ereignis kann die Daten aus der `clipboard`-Eigenschaft des Ereignisobjekts extrahieren.

Wenn die Zwischenablage ein von der Anwendung definiertes Format enthält, bestimmt der Parameter `transferMode`, der an die Methode `getData()` des Clipboard-Objekts übergeben wurde, ob der Ziehmanager einen Verweis oder eine serialisierte Version des Objekts ausgibt.

Das folgende Beispiel präsentiert eine Ereignisprozedur für das `nativeDragDrop`-Ereignis.

```
 import flash.desktop.Clipboard;
import flash.events.NativeDragEvent;

public function onDrop(event:NativeDragEvent):void {
    if (event.clipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)) {
    var text:String =
        String(event.clipboard.getData(ClipboardFormats.TEXT_FORMAT,
                                    ClipboardTransferMode.ORIGINAL_PREFERRED));
}
```

Ist die Ereignisprozedur einmal vorhanden, ist das Clipboard-Objekt nicht mehr gültig. Bei jedem Versuch, auf das Objekt oder seine Daten zuzugreifen, kommt es zu einen Fehler.

Aktualisieren der visuellen Darstellung einer Komponente

Adobe AIR 1.0 und höher

Eine Komponente kann seine visuelle Darstellung aufgrund des NativeDragEvent-Ereignisses aktualisieren. In der folgenden Tabelle ist beschrieben, welche Art von Änderung eine typische Komponente als Reaktion auf verschiedene Ereignisse vornimmt:

Ereignis	Beschreibung
nativeDragStart	Das initiierende, interaktive Objekt kann mithilfe des `nativeDragStart`-Ereignisses visuell anzeigen, dass es die Ziehbewegung ausgelöst hat.
nativeDragUpdate	Das initiierende, interaktive Objekt kann mithilfe des nativeDragUpdate-Ereignisses seinen Status während der Bewegung aktualisieren. (Dieses Ereignis gibt es nicht in AIR für Linux.)
nativeDragEnter	Ein potentiell empfangendes, interaktives Objekt kann mithilfe dieses Ereignisses den Fokus zu übernehmen oder visuell anzeigen, ob es das abzulegende Element akzeptieren kann oder nicht.
nativeDragOver	Ein potentiell empfangendes, interaktives Objekt kann dieses Ereignis dazu verwenden, mit einem interaktiven Objekt auf die Mausbewegung zu reagieren, etwa wenn der Mauscursor über den „sensiblen“ Bereich einer komplexen Komponente geführt wird, z. B. eine Straßenkarte.
nativeDragExit	Ein potentiell empfangendes, interaktives Objekt kann dieses Ereignis dazu verwenden, seinen vorigen Status wieder herzustellen, wenn eine Ziehbewegung die Grenzen dieses Objekts verlässt.
nativeDragComplete	Das initiierende, interaktive Objekt kann mithilfe dieses Ereignisses das verbundene Datenmodell aktualisieren, z. B. durch Entfernen eines Elements aus einer Liste oder durch Wiederherstellen seines visuellen Status.

Verfolgen der Mausposition beim Hineinziehen

Adobe AIR 1.0 und höher

Während sich eine Ziehbewegung über einer interaktiven Komponente befindet, löst diese `nativeDragOver`-Ereignisse aus. Diese Ereignisse werden alle paar Millisekunden sowie bei jeder Mausbewegung ausgelöst. Anhand des `nativeDragOver`-Ereignisobjekts lässt sich die Position des Mauscursors über der Komponente bestimmen. Dieser Zugriff auf die Mausposition kann bei komplexen, empfangenden Komponenten, die sich nicht aus Unterkomponenten zusammensetzen, sehr nützlich sein. Angenommen, in Ihrer Anwendung wird eine Bitmap mit einer Straßenkarte angezeigt, und Sie möchten bestimmte Bereiche auf der Karte hervorheben, wenn der Benutzer Informationen daraufzieht. Dafür können Sie die Mauskoordinaten nutzen, die vom `nativeDragOver`-Ereignis gemeldet werden, und somit die Mausposition innerhalb der Karte bestimmen.

Drag & Drop in HTML

Adobe AIR 1.0 und höher

Um Daten in eine HTML-gestützte Anwendung zu ziehen oder aus einer HTML-Anwendung herauszuziehen (oder in den HTML-Code eines HTML-Loaders zu ziehen oder aus dem HTML-Loader herauszuziehen), gibt es spezielle HTML-Drag & Drop-Ereignisse. Die HTML-Drag & Drop-API ermöglicht es Ihnen, Drag & Drop-Aktionen in und aus DOM-Elementen im HTML-Inhalt durchzuführen.

Hinweis: *Sie können auch die AIR-APIs „NativeDragEvent“ und „NativeDragManager“ verwenden. Dazu definieren Sie einen Listener für Ereignisse des HTMLLoader-Objekts, das den HTML-Inhalt enthält. Allerdings ist die HTML-API besser in das HTML-DOM integriert und bietet Ihnen Möglichkeiten zur Steuerung von Standardverhalten.*

Standardverhalten bei Drag & Drop-Aktionen

Adobe AIR 1.0 und höher

Die HTML-Umgebung stellt das Standardverhalten für Drag & Drop-Bewegungen, einschließlich Text, Bildern und URLs, zur Verfügung. Wenn Sie das Standardverhalten verwenden, können Sie Daten mit diesen Datentypen immer aus einem Element herausziehen. Sie können allerdings nur Text in ein Element ziehen und nur unter der Voraussetzung, das Element befindet sich in einem editierbaren Bereich einer Seite. Wenn Sie Text innerhalb eines editierbaren Bereichs einer Seite oder auch in einen anderen editierbaren Bereich derselben Seite ziehen, wird als Standardverhalten eine Verschiebaktion durchgeführt. Wenn Sie Text von einem nicht editierbaren Bereich oder von außerhalb der Anwendung in einen editierbaren Bereich ziehen, wird als Standardverhalten eine Kopieraktion durchgeführt.

Sie können das Standardverhalten überschreiben, indem Sie die Drag & Drop-Ereignisse selbst definieren. Um das Standardverhalten zu deaktivieren, rufen Sie die `preventDefault()`-Methode der Objekte auf, die für die Drag & Drop-Ereignisse ausgelöst wurden. Anschließend können Sie nach Bedarf Daten in das Ablageziel einfügen oder Daten aus der Ziehquelle entfernen.

Standardmäßig kann der Benutzer jeglichen Text sowie Bilder und Links auswählen und ziehen. Mithilfe der WebKit-CSS-Eigenschaft `-webkit-user-select` können Sie steuern, wie ein HTML-Element ausgewählt werden kann. Wenn Sie zum Beispiel `-webkit-user-select` auf `none` setzen, ist der Elementinhalt nicht auswählbar und kann somit auch nicht gezogen werden. Und mithilfe der CSS-Eigenschaft `-webkit-user-drag` können Sie festlegen, ob ein Element als Ganzes gezogen werden kann. Allerdings wird der Inhalt des Elements dann gesondert verarbeitet. Der Benutzer könnte dann nach wie vor ausgewählte Textpassagen ziehen. Weitere Informationen finden Sie unter „CSS in AIR“ auf Seite 1042.

Drag & Drop-Ereignisse in HTML

Adobe AIR 1.0 und höher

Im Initiatorelement, also dort, wo die Ziehaktion beginnt, werden folgende Ereignisse ausgelöst:

Ereignis	Beschreibung
dragstart	Wird ausgelöst, wenn der Benutzer eine Ziehbewegung beginnt. Die Ereignisprozedur für dieses Ereignis kann das Ziehen verhindern, falls notwendig, indem sie die preventDefault()-Methode des Ereignisobjekts aufruft. Um zu steuern, ob die zu ziehenden Daten kopiert, verschoben oder verknüpft werden sollen, setzen Sie die Eigenschaft „effectAllowed“. Ausgewählter Text, Bilder und Links werden gemäß dem Standardverhalten in die Zwischenablage gelegt, Sie können aber mithilfe der dataTransfer-Eigenschaft des Ereignisobjekts andere Daten für die Ziehbewegung festlegen.
drag	Wird während der Ziehbewegung kontinuierlich ausgelöst.
dragend	Wird ausgelöst, wenn der Benutzer die Maustaste loslässt, um die Ziehbewegung zu beenden.

Vom Ablageziel werden folgende Ereignisse ausgelöst:

Ereignis	Beschreibung
dragover	Wird kontinuierlich ausgelöst, solange die Ziehbewegung innerhalb der Elementgrenzen bleibt. Die Prozedur für dieses Ereignis sollte die Eigenschaft „dataTransfer.dropEffect“ setzen, um festzulegen, ob nach dem Loslassen der Maus eine Kopier-, Verschieb- oder Verknüpfaktion ausgeführt wird.
dragenter	Wird ausgelöst, wenn eine Ziehbewegung über die Begrenzung eines Elements geführt wird. Wenn Sie in einer dragenter-Ereignisprozedur Eigenschaften des Objekts „dataTransfer“ ändern, werden diese Änderungen bald darauf von dem nächsten dragover-Ereignis überschrieben. Andererseits gibt es zwischen einem dragenter- und dem ersten dragover-Ereignis eine kurze Verzögerung. Dies kann zur Folge haben, dass der Cursor aufblinkt, wenn verschiedene Eigenschaften gesetzt wurden. In vielen Fällen können Sie für beide Ereignisse dieselbe Ereignisprozedur verwenden.
dragleave	Wird ausgelöst, wenn die Ziehbewegung die Elementgrenzen verlässt.
drop	Wird ausgelöst, wenn der Benutzer die Daten auf dem Element ablegt. Auf die gezogenen Daten kann nur innerhalb der Prozedur für dieses Ereignis zugegriffen werden.

Das als Reaktion auf diese Ereignisse ausgelöste Ereignisobjekt ähnelt einem Mausereignis. Sie können mithilfe von Mausereigniseigenschaften wie (`clientX`, `clientY`) und (`screenX`, `screenY`) die Mausposition bestimmen.

Die wichtigste Eigenschaft eines Ziehereignisobjekts ist `dataTransfer`, das die gezogenen Daten enthält. Das `dataTransfer`-Objekt selbst verfügt über folgende Eigenschaften und Methoden:

Eigenschaft oder Methode	Beschreibung
effectAllowed	Der von der Quelle des Ziehvorgangs zugelassene Effekt. Typischerweise setzt die Prozedur für das dragstart-Ereignis diesen Wert. (Siehe „Zieheffekte in HTML“ auf Seite 659.)
dropEffect	Der vom Ziel oder Benutzer gewählte Effekt. Wenn Sie den Ablegeffekt (`dropEffect`) in einer `dragover`- oder `dragenter`-Ereignisprozedur setzen, aktualisiert AIR den Mauscursor. Dadurch zeigt der Curser immmer den Effekt an, der beim Loslassen der Maustaste eintritt. Wenn der eingestellte `dropEffect` keinem der zulässigen Effekte entspricht, kann das Element nicht abgelegt werden und der *Nicht-verfügbar*-Cursor wird angezeigt. Wenn Sie keinen `dropEffect` als Reaktion auf das letzte `dragover`- oder `dragenter`-Ereignis gesetzt haben, kann der Benutzer mithilfe der Standardzusatztasten des jeweiligen Betriebssystems aus den zulässigen Effekten einen auswählen. Der Abschlusseffekt wird von der `dropEffect`-Eigenschaft des Objekts gemeldet, das für `dragend` ausgelöst wurde. Wenn der Benutzer den Ablegvorgang abbricht, indem er die Maustaste außerhalb eines zulässigen Ziels loslässt, wird `dropEffect` auf den Wert `none` gesetzt.
types	Ein Array mit den MIME-Typen aller gegenwärtigen Datenformate im Objekt `dataTransfer`.
getData(mimeType)	Ruft die Daten in dem vom `mimeType`-Parameter angegebenen Format ab. Die Methode `getData()` kann nur als Reaktion auf das Ereignis `drop` aufgerufen werden.
setData(mimeType)	Fügt die Daten in dem vom `mimeType`-Parameter angegebenen Format in das dataTransfer-Objekt ein. Sie können Daten in mehreren Formaten einbinden, indem Sie `setData()` für jeden MIME-Typ aufrufen. Daten, die vom Standardziehverhalten in das Objekt `dataTransfer` eingefügt wurden, werden gelöscht. Die Methode `setData()` kann nur als Reaktion auf das Ereignis `dragstart` aufgerufen werden.
clearData(mimeType)	Löscht alle Daten in dem vom `mimeType`-Parameter angegebenen Format ab.
setDragImage(image, offsetX, offsetY)	Definiert ein benutzerdefiniertes Ziehbild. Die `setDragImage()`-Methode kann nur in Reaktion auf das dragstart-Ereignis aufgerufen werden und nur, wenn ein gesamtes HTML-Element gezogen wird, indem sein `-webkit-user-drag`-CSS-Stil auf `element` gesetzt wird. Der `image`-Parameter kann ein JavaScript-Element oder Image-Objekt sein.

MIME-Typen für HTML-Drag & Drop-Aktionen

Adobe AIR 1.0 und höher

Für das `dataTransfer`-Objekt eines HTML-Drag & Drop-Ereignisses können folgende MIME-Typen verwendet werden:

Datenformat	MIME-Typ
Text	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
File list	"application/x-vnd.adobe.air.file-list"

Sie können auch andere MIME-Strings verwenden, auch von der Anwendung definierte Strings. Es ist jedoch möglich, dass andere Anwendungen die übertragenen Daten dann nicht erkennen oder verarbeiten können. Es liegt in Ihrer Verantwortung, Daten im erwarteten Format in das `dataTransfer`-Objekt einzufügen.

Wichtig: *Nur Code, der in der Anwendungs-Sandbox ausgeführt wird, kann auf abgelegte Dateien zugreifen. Der Versuch, Eigenschaften eines File-Objekts innerhalb einer anwendungsfremden Sandbox zu lesen oder zu setzen, erzeugt einen Sicherheitsfehler. Weitere Informationen finden Sie unter „Abgelegte Dateien in anwendungsfremden HTML-Sandboxen“ auf Seite 663.*

Zieheffekte in HTML

Adobe AIR 1.0 und höher

Der Initiator der Ziehbewegung kann die zugelassenen Zieheffekte einschränken, indem er die Eigenschaft `dataTransfer.effectAllowed` in der Prozedur für das `dragstart`-Ereignis setzt. Es können die folgenden Stringwerte verwendet werden:

Stringwert	Beschreibung
"none"	Es sind keine Ziehoperationen zugelassen.
"copy"	Die Daten werden in das Ziel kopiert, bleiben aber auch am Herkunftsort erhalten.
"link"	Die Daten werden gemeinsam mit dem Ziel genutzt, indem eine Verknüpfung zum Herkunftsort erstellt wird.
"move”	Die Daten werden vom Herkunftsort entfernt und in das Ziel kopiert.
"copyLink"	Die Daten können kopiert oder verknüpft werden.
"copyMove"	Die Daten können kopiert oder verschoben werden.
"linkMove"	Die Daten können verknüpft oder verschoben werden.
"all"	Die Daten können kopiert, verschoben oder verknüpft werden. *all* ist der Standardeffekt, wenn Sie das Standardverhalten deaktiviert haben.

Vom Ziel der Ziehbewegung kann in der Eigenschaft `dataTransfer.dropEffect` festgelegt werden, welche Aktion durchgeführt wird, wenn der Benutzer den Ablegvorgang abschließt. Ist der Ablegeffekt eine der zulässigen Aktionen, zeigt das System den entsprechenden Kopier-, Verschieb- oder Verknüpfcursor an. Andernfalls zeigt das System den *Nicht-verfügbar*-Cursor an. Wenn vom Ziel kein Ablegeffekt eingestellt wurde, kann der Benutzer mithilfe der Zusatztasten aus den zulässigen Aktionen einen Effekt auswählen.

Den `dropEffect`-Wert legen Sie in den Prozeduren für die Ereignisse `dragover` und `dragenter` fest:

```
function doDragStart(event) {
    event.dataTransfer.setData("text/plain","Text to drag");
    event.dataTransfer.effectAllowed = "copyMove";
}

function doDragOver(event) {
    event.dataTransfer.dropEffect = "copy";
}

function doDragEnter(event) {
    event.dataTransfer.dropEffect = "copy";
}
```

Hinweis: *Obwohl Sie die Eigenschaft `dropEffect` in der Prozedur für `dragenter` immer einstellen sollten, müssen Sie sich im Klaren darüber sein, dass das nächste `dragover`-Ereignis die Eigenschaft auf ihren Standardwert zurücksetzt. Legen Sie daher als Reaktion auf beide Ereignisse einen `dropEffect`-Wert fest.*

Herausziehen von Daten aus einem HTML-Element

Adobe AIR 1.0 und höher

Gemäß dem Standardverhalten werden die meisten Inhalte auf einer HTML-Seite beim Ziehen kopiert. Mit den CSS-Eigenschaften `-webkit-user-select` und `-webkit-user-drag` können Sie steuern, welche Inhalte gezogen werden dürfen.

Überschreiben Sie das Herauszieh-Standardverhalten in der Prozedur für das Ereignis `dragstart`. Rufen Sie die Methode `setData()` der Eigenschaft `dataTransfer` des Ereignisobjekts auf, um Ihre eigenen Daten in die Ziehbewegung aufzunehmen.

Um anzuzeigen, welche Zieheffekte ein Quellobjekt unterstützt, wenn Sie sich auf das Standardverhalten verlassen, stellen Sie die Eigenschaft `dataTransfer.effectAllowed` des Ereignisobjekts ein, das für das Ereignis `dragstart` ausgelöst wurde. Sie können jede beliebige Kombination an Effekten verwenden. Wenn beispielsweise ein Quellelement sowohl die *Kopier*- als auch die *Verknüpf*effekte unterstützt, setzen Sie die Eigenschaft auf `"copyLink"`.

Festlegen der Ziehdaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Fügen Sie mithilfe der Eigenschaft `dataTransfer` die Daten für die Ziehbewegung in die Prozedur für das Ereignis `dragstart` ein. Fügen Sie mithilfe der Methode `dataTransfer.setData()` Daten in die Zwischenablage ein und übergeben Sie dabei den MIME-Typ und die zu übertragenden Daten.

Angenommen, Sie haben in einer Anwendung ein Bildelement mit der ID *imageOfGeorge*, dann könnten Sie die folgende dragstart-Ereignisprozedur verwenden. In dem Beispiel werden Bilder von George in verschiedenen Datenformaten eingefügt, sodass andere Anwendungen die gezogenen Daten mit größerer Wahrscheinlichkeit nutzen können.

```
function dragStartHandler(event){
    event.dataTransfer.effectAllowed = "copy";

    var dragImage = document.getElementById("imageOfGeorge");
    var dragFile = new air.File(dragImage.src);
    event.dataTransfer.setData("text/plain","A picture of George");
    event.dataTransfer.setData("image/x-vnd.adobe.air.bitmap", dragImage);
    event.dataTransfer.setData("application/x-vnd.adobe.air.file-list",
                               new Array(dragFile));
}
```

Hinweis: *Wenn Sie die Methode `setData()` der Eigenschaft `dataTransfer` aufrufen, werden vom Standard-Drag & Drop-Verhalten keine Daten hinzugefügt.*

Hineinziehen von Daten in ein HTML-Element

Adobe AIR 1.0 und höher

Gemäß dem Standardverhalten kann nur Text in die editierbaren Bereiche einer Seite eingefügt werden. Sie können festlegen, dass ein HTML-Element und die ihm untergeordneten Elemente editierbar sein sollen, indem Sie in das öffnende Tag das Attribut `contenteditable` einfügen. Sie können auch ein ganzes Dokument editierbar machen, indem Sie die Eigenschaft `designMode` des document-Objekts auf `"on"` setzen.

Um auf einer Seite ein alternatives Verhalten für das Hineinziehen zu definieren, verwenden Sie für alle Elemente, die gezogene Daten akzeptieren können, die Ereignisse `dragenter`, `dragover` und `drop`.

Aktivieren des Hineinziehens

Adobe AIR 1.0 und höher

Um die Hineinziehbewegung nach eigener Maßgabe zu verarbeiten, müssen Sie zuerst das Standardverhalten deaktivieren. Definieren Sie für alle HTML-Elemente, die Sie als Ablageziele verwenden wollen, Listener für die Ereignisse `dragenter` und `dragover`. Rufen Sie in den Prozeduren für diese Ereignisse die `preventDefault()`-Methode des ausgelösten Ereignisobjekts auf. Durch ein solches Deaktivieren des Standardverhaltens können auch nicht-editierbare Bereich ein abzulegendes Element aufnehmen.

Abrufen der abgelegten Daten

Adobe AIR 1.0 und höher

In der Prozedur für das Ereignis `ondrop` können Sie auf die abgelegten Daten zugreifen:

```
function doDrop(event){
    droppedText = event.dataTransfer.getData("text/plain");
}
```

Lesen Sie mithilfe der Methode `dataTransfer.setData()` Daten in die Zwischenablage ein und übergeben Sie dabei den MIME-Typ des zu lesenden Datenformats. Mithilfe der `types`-Eigenschaft des `dataTransfer`-Objekts können Sie herausfinden, welche Datenformate verfügbar sind. Der `types`-Array enthält den MIME-Typ-String für jedes verfügbare Format.

Wenn Sie das Standardverhalten in dem Ereignis „dragenter“ oder „dragover“ deaktivieren, sind Sie selbst dafür verantwortlich, dass abgelegte Daten an der richtigen Stelle im Dokument platziert werden. Es gibt keine API, um eine Mausposition innerhalb eines Elements in eine Einfügemarke zu konvertieren. Diese Einschränkung macht es mitunter schwierig, Ziehbewegungen für das Einfügen zu implementieren.

Beispiel: Überschreiben des Standardverhaltens für das Hineinziehen von Elementen in HTML-Elemente

Adobe AIR 1.0 und höher

In diesem Beispiel wird ein Ablageziel implementiert, das eine Tabelle mit allen im abgelegten Element verfügbaren Datenformaten anzeigt.

Das Standardverhalten wird verwendet, um Text, Links und Bilder in die Anwendung zu ziehen. In dem Beispiel wird für das Element „div“, das als Ablageziel dient, das Standardverhalten für das Hineinziehen deaktiviert. Wenn Sie nicht editierbare Inhalte so einrichten wollen, dass sie eine Hineinziehbewegung akzeptieren, rufen Sie die Methode `preventDefault()` des Ereignisobjekts auf, das sowohl für das Ereignis `dragenter` als auch für das Ereignis `dragover` ausgelöst wurde. Als Reaktion auf das Ereignis `drop` konvertiert die Prozedur die übertragenen Daten in ein row-HTML-Element und fügt diese Zeile in die anzuzeigende Tabelle ein.

```
 <html>
<head>
<title>Drag-and-drop</title>
<script language="javascript" type="text/javascript" src="AIRAliases.js"></script>
<script language="javascript">
    function init(){
        var target = document.getElementById('target');
        target.addEventListener("dragenter", dragEnterOverHandler);
        target.addEventListener("dragover", dragEnterOverHandler);
        target.addEventListener("drop", dropHandler);

        var source = document.getElementById('source');
        source.addEventListener("dragstart", dragStartHandler);
        source.addEventListener("dragend", dragEndHandler);

        emptyRow = document.getElementById("emptyTargetRow");
    }

    function dragStartHandler(event){
        event.dataTransfer.effectAllowed = "copy";
    }

    function dragEndHandler(event){
        air.trace(event.type + ": " + event.dataTransfer.dropEffect);
    }

    function dragEnterOverHandler(event){
        event.preventDefault();
    }

    var emptyRow;
    function dropHandler(event){
        for(var prop in event){
            air.trace(prop + " = " + event[prop]);
        }
        var row = document.createElement('tr');
        row.innerHTML = "<td>" + event.dataTransfer.getData("text/plain") + "</td>" +
            "<td>" + event.dataTransfer.getData("text/html") + "</td>" +
            "<td>" + event.dataTransfer.getData("text/uri-list") + "</td>" +
            "<td>" + event.dataTransfer.getData("application/x-vnd.adobe.air.file-list") +
            "</td>";

        var imageCell = document.createElement('td');
        if((event.dataTransfer.types.toString()).search("image/x-vnd.adobe.air.bitmap") > -
1){
            imageCell.appendChild(event.dataTransfer.getData("image/x-
vnd.adobe.air.bitmap"));
        }
        row.appendChild(imageCell);
```

```
        var parent = emptyRow.parentNode;
        parent.insertBefore(row, emptyRow);
    }
</script>
</head>
<body onLoad="init()" style="padding:5px">
<div>
    <h1>Source</h1>
       <p>Items to drag:</p>
    <ul id="source">
        <li>Plain text.</li>
           <li>HTML <b>formatted</b> text.</li>
           <li>A <a href="http://www.adobe.com">URL.</a></li>
           <li><img src="icons/AIRApp_16.png" alt="An image"/></li>
           <li style="-webkit-user-drag:none;">
        Uses "-webkit-user-drag:none" style.
           </li>
           <li style="-webkit-user-select:none;">
        Uses "-webkit-user-select:none" style.
           </li>

       </ul>
</div>
<div id="target" style="border-style:dashed;">
    <h1 >Target</h1>
       <p>Drag items from the source list (or elsewhere).</p>
       <table id="displayTable" border="1">
        <tr><th>Plain text</th><th>Html text</th><th>URL</th><th>File list</th><th>Bitmap
Data</th></tr>
           <tr
id="emptyTargetRow"><td> </td><td> </td><td> </td><td> </td><td> </
td></tr>
       </table>
       </div>
</div>
</body>
</html>
```

Abgelegte Dateien in anwendungsfremden HTML-Sandboxen

Adobe AIR 1.0 und höher

Anwendungsfremde Inhalte können nicht auf die File-Objekte zugreifen, die entstehen, wenn Dateien in eine AIR-Anwendung gezogen werden. Es ist auch nicht möglich, eines dieser File-Objekte über eine Sandbox-Brücke an Anwendungsinhalte zu übergeben. (Auf die Objekteigenschaften muss während der Serialisierung zugegriffen werden.) Sie können dennoch Dateien in Ihrer Anwendung ablegen, indem Sie für die nativeDragDrop-AIR-Ereignisse des HTMLLoader-Objekts Listener einrichten.

Wenn ein Benutzer normalerweise eine Datei in einem Bild ablegt, das anwendungsfremde Inhalte beherbergt, wird das drop-Ereignis nicht vom untergeordneten an das übergeordnete Objekt weitergegeben. Da jedoch die vom HTMLLoader (dem Container für alle HTML-Inhalte in einer AIR-Anwendung) ausgelösten Ereignisse nicht Teil des HTML-Ereignisflusses sind, können Sie das drop-Ereignis im Anwendungsinhalt immer noch empfangen.

Um das Ereignis für eine abzulegende Datei zu empfangen, definiert das übergeordnete Dokument für den HTMLLoader einen Ereignis-Listener, und zwar mithilfe der vom `window.htmlLoader` bereitgestellten Verweis:

```
window.htmlLoader.addEventListener("nativeDragDrop",function(event){
    var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
    air.trace(filelist[0].url);
});
```

Im folgenden Beispiel wird ein übergeordnetes Dokument verwendet, das eine untergeordnete Seite in eine Remote-Sandbox (http://localhost/) lädt. Das übergeordnete Dokument wartet auf das Ereignis `nativeDragDrop` des HTMLLoader-Objekts und ermittelt die Datei-URL.

```
<html>
<head>
<title>Drag-and-drop in a remote sandbox</title>
<script language="javascript" type="text/javascript" src="AIRAliases.js"></script>
<script language="javascript">
    window.htmlLoader.addEventListener("nativeDragDrop",function(event){
        var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
        air.trace(filelist[0].url);
    });
</script>
</head>
<body>
       <iframe src="child.html"
               sandboxRoot="http://localhost/"
               documentRoot="app:/"
               frameBorder="0"  width="100%" height="100%">
       </iframe>
</body>
</html>
```

Das untergeordnete Dokument muss ein gültiges Ablageziel bereitstellen, indem es die Ereignisobjektmethode `preventDefault()` in den HTML-Ereignisprozeduren `dragenter` und `dragover` aufruft. Andernfalls kann kein drop-Ereignis eintreten.

```
<html>
<head>
    <title>Drag and drop target</title>
    <script language="javascript" type="text/javascript">
        function preventDefault(event){
            event.preventDefault();
        }
    </script>
</head>
<body ondragenter="preventDefault(event)" ondragover="preventDefault(event)">
<div>
<h1>Drop Files Here</h1>
</div>
</body>
</html>
```

Ablegen von Dateizusagen

Adobe AIR 2 und höher

Eine Dateizusage ist ein Drag-and-Drop-Zwischenablageformat, das es einem Benutzer ermöglicht, eine Datei, die noch nicht vorhanden ist, aus einer AIR-Anwendung zu ziehen. Mithilfe von Dateizusagen ist es in einer Anwendung beispielsweise möglich, ein Proxysymbol in einen Ordner auf dem Desktop zu ziehen. Das Proxysymbol repräsentiert eine Datei oder bestimmte Daten, von denen bekannt ist, dass sie an einer URL zur Verfügung stehen. Nachdem der Anwender das Symbol abgelegt hat, lädt die Laufzeit die Daten herunter und schreibt die Datei an den Ablageort.

Mithilfe der URLFilePromise-Klasse in einer AIR-Anwendung können Dateien, die an einer URL verfügbar sind, per Drag & Drop gezogen und abgelegt werden. Die URLFilePromise-Implementierung steht in der aircore-Bibliothek im Rahmen des AIR 2 SDK zur Verfügung. Verwenden Sie entweder die Datei „aircore.swc“ oder „aircore.swf“ im SDK-Verzeichnis „frameworks/libs/air“.

Stattdessen können Sie auch Ihre eigene Logik für Dateizusagen mit der IFilePromise-Schnittstelle implementieren (diese Schnittstelle ist im flash.desktop-Laufzeitpaket definiert).

Dateizusagen ähneln konzeptuell dem verzögerten Rendering unter Verwendung einer Datenprozedurfunktion in der Zwischenablage. Verwenden Sie beim Ziehen und Ablegen von Dateien Dateizusagen anstelle des verzögerten Rendering. Das verzögerte Rendering kann bei Ziehbewegungen zu unerwünschten Pausen führen, während die Daten generiert oder heruntergeladen werden. Verwenden Sie das verzögerte Rendering zum Kopieren und Einfügen (für diesen Zweck werden Dateizusagen nicht unterstützt).

Einschränkungen bei Verwendung von Dateizusagen

Bei Dateizusagen gelten die folgenden Einschränkungen im Vergleich mit anderen Datenformaten, die per Drag & Drop in die Zwischenablage platziert werden können:

- Dateizusagen können nur aus einer AIR-Anwendung gezogen werden, nicht jedoch in einer AIR-Anwendung abgelegt werden.
- Dateizusagen werden nicht unter allen Betriebssystemen unterstützt. Mit der `Clipboard.supportsFilePromise`-Eigenschaft können Sie testen, ob Dateizusagen auf dem Hostsystem unterstützt werden. Auf Systemen, die Dateizusagen nicht unterstützen, sollten Sie einen Alternativmechanismus zum Herunterladen oder Generieren der Dateidaten bereitstellen.
- Dateizusagen können nicht mit der Zwischenablage für das Kopieren und Einfügen verwendet werden (`Clipboard.generalClipboard`).

Verwandte Hilfethemen

flash.desktop.IFilePromise

air.desktop.URLFilePromise

Ablegen von Remote-Dateien

Adobe AIR 2 und höher

Verwenden Sie die URLFilePromise-Klasse zum Erstellen von Dateizusagenobjekten für Dateien oder Daten, die an einer URL zur Verfügung stehen. Fügen Sie der Zwischenablage mit dem Zwischenablageformat `FILE_PROMISE_LIST` mindestens ein Dateizusagenobjekt hinzu. Im folgenden Beispiel wird eine einzelne Datei, die unter http://www.example.com/foo.txt zur Verfügung steht, heruntergeladen und als „bar.txt" am Ablageort gespeichert. (Die Name der Remote-Datei muss nicht mit dem Namen der lokalen Datei übereinstimmen.)

```
if( Clipboard.supportsFilePromise )
{
    var filePromise:URLFilePromise = new URLFilePromise();
    filePromise.request = new URLRequest("http://example.com/foo.txt");
    filePromise.relativePath = "bar.txt";

    var fileList:Array = new Array( filePromise );
    var clipboard:Clipboard = new Clipboard();
    clipboard.setData( ClipboardFormats.FILE_PROMISE_LIST_FORMAT, fileList );
    NativeDragManager.doDrag( dragSource, clipboard );
}
```

Sie können dem Anwender die Möglichkeit geben, mehrere Dateien gleichzeitig zu ziehen; dazu fügen Sie dem Array, das der Zwischenablage zugewiesen ist, mehrere Dateizusagenobjekte hinzu. Außerdem können Sie für die `relativePath`-Eigenschaft Unterverzeichnisse angeben, damit einige oder alle Dateien in einem Unterordner relativ zum Ablageort platziert werden.

Im folgenden Beispiel wird gezeigt, wie eine Ziehoperation mit mehreren Dateizusagen eingeleitet wird. In diesem Beispiel wird die HTML-Seite *article.html* zusammen mit ihren beiden verknüpften Bilddateien als Dateizusage in die Zwischenablage platziert. Die Bilder werden in den Unterordner *images* kopiert, sodass die relativen Hyperlinks beibehalten werden.

```
if( Clipboard.supportsFilePromise )
{   //Create the promise objects
    var filePromise:URLFilePromise = new URLFilePromise();
    filePromise.request = new URLRequest("http://example.com/article.html");
    filePromise.relativePath = "article.html";

    var image1Promise:URLFilePromise = new URLFilePromise();
    image1Promise.request = new URLRequest("http://example.com/images/img_1.jpg");
    image1Promise.relativePath = "images/img_1.html";
    var image2Promise:URLFilePromise = new URLFilePromise();
    image2Promise.request = new URLRequest("http://example.com/images/img_2.jpg");
    image2Promise.relativePath = "images/img_2.jpg";


    //Put the promise objects onto the clipboard inside an array
    var fileList:Array = new Array( filePromise, image1Promise, image2Promise );
    var clipboard:Clipboard = new Clipboard();
    clipboard.setData( ClipboardFormats.FILE_PROMISE_LIST_FORMAT, fileList );
    //Start the drag operation
    NativeDragManager.doDrag( dragSource, clipboard );
}
```

Implementieren der IFilePromise-Schnittstelle

Adobe AIR 2 und höher

Zum Bereitstellen von Dateizusagen für Ressourcen, die sich nicht über ein URLFilePromise-Objekt aufrufen lassen, können Sie die IFilePromise-Schnittstelle in einer benutzerdefinierten Klasse implementieren. Die IFilePromise-Schnittstelle definiert die Methoden und Eigenschaften, die die AIR-Laufzeit für den Zugriff auf die Daten verwendet, die in eine Datei geschrieben werden sollen, wenn die Dateizusage abgelegt wird.

Eine IFilePromise-Implementierung übergibt ein anderes Objekt an die AIR-Laufzeit, das die Daten für die Dateizusage bereitstellt. Dieses Objekt muss die IDataInput-Schnittstelle implementieren, die von der AIR-Laufzeit zum Lesen der Daten verwendet wird. Beispielsweise verwendet die URLFilePromise-Klasse, die IFilePromise implementiert, ein URLStream-Objekt als Datenprovider.

AIR kann die Daten synchron oder asynchron lesen. Die IFilePromise-Implementierung meldet den unterstützten Zugriffsmodus durch Rückgabe des entsprechenden Wertes in der `isAsync`-Eigenschaft. Für den asynchronen Datenzugriff muss das Datenprovider-Objekt die IEventDispatcher-Schnittstelle implementieren und die erforderlichen Ereignisse auslösen, wie `open`, `progress` und `complete`.

Als Datenprovider für eine Dateizusage können Sie eine benutzerdefinierte Klasse oder eine der folgenden integrierten Klassen verwenden:

- ByteArray (synchron)
- FileStream (synchron oder asynchron)
- Socket (asynchron)
- URLStream (asynchron)

Zum Implementieren der IFilePromise-Schnittstelle müssen Sie Code für die folgenden Funktionen und Eigenschaften angeben:

- `open():IDataInput` – Gibt das Datenprovider-Objekt zurück, aus dem die Daten für die zugesagte Datei gelesen werden. Das Objekt muss die IDataInput-Schnittstelle implementieren. Wenn die Daten asynchron bereitgestellt werden, muss das Objekt auch die IEventDispatcher-Schnittstelle implementieren und die erforderlichen Ereignisse auslösen (siehe „Verwenden eines asynchronen Datenprovider in einer Dateizusage“ auf Seite 669).
- `get relativePath():String` – Gibt den Pfad, einschließlich des Namens, für die erstellte Datei an. Dieser Pfad wird relativ zu dem Ablageort aufgelöst, den der Anwender beim Ziehen und Ablegen ausgewählt hat. Um sicherzustellen, dass der Pfad das passende Trennzeichen für das Host-Betriebssystem verwendet, geben Sie die `File.separator`-Konstante an, wenn Sie Pfade festlegen, die Verzeichnisse enthalten. Sie können eine Set-Funktion hinzufügen oder einen Konstruktorparameter verwenden, damit der Pfad zur Laufzeit festgelegt werden kann.
- `get isAsync():Boolean` – Informiert die AIR-Laufzeit, ob das Datenprovider-Objekt die Daten synchron oder asynchron bereitstellt.
- `close():void` – Wird von der Laufzeit aufgerufen, wenn die Daten vollständig gelesen wurden (oder wenn ein Fehler verhindert, dass die Daten weiter gelesen werden können). Diese Funktion kann zur Ressourcenbereinigung verwendet werden.
- `reportError( e:ErrorEvent ):void` – Wird von der Laufzeit aufgerufen, wenn beim Lesen der Daten ein Fehler auftritt.

Alle IFilePromise-Methoden werden während einer Drag & Drop-Operation, die die Dateizusage umfasst, von der Laufzeit aufgerufen. Normalerweise sollte die Anwendungslogik keine dieser Methoden direkt aufrufen.

Verwenden eines synchronen Datenprovider in einer Dateizusage

Adobe AIR 2 und höher

Die einfachste Methode zur Implementierung der IFilePromise-Schnittstelle ist die Verwendung eines synchronen Datenprovider-Objekts, wie ein ByteArray-Objekt oder ein synchrones FileStream-Objekt. Im folgenden Beispiel wird ein ByteArray-Objekt erstellt, mit Daten gefüllt und dann beim Aufruf der `open()`-Methode zurückgegeben.

```
package
{
    import flash.desktop.IFilePromise;
    import flash.events.ErrorEvent;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;

    public class SynchronousFilePromise implements IFilePromise
    {
        private const fileSize:int = 5000; //size of file data
        private var filePath:String = "SynchronousFile.txt";

        public function get relativePath():String
        {
            return filePath;
        }

        public function get isAsync():Boolean
        {
            return false;
        }

        public function open():IDataInput
        {
            var fileContents:ByteArray = new ByteArray();

            //Create some arbitrary data for the file
            for( var i:int = 0; i < fileSize; i++ )
            {
                fileContents.writeUTFBytes( 'S' );
            }

            //Important: the ByteArray is read from the current position
            fileContents.position = 0;
            return fileContents;
        }

        public function close():void
        {
            //Nothing needs to be closed in this case.
        }

        public function reportError(e:ErrorEvent):void
        {
            trace("Something went wrong: " + e.errorID + " - " + e.type + ", " + e.text );
        }
    }
}
```

In der Praxis sind synchrone Dateizusagen nur von eingeschränktem Nutzen. Bei einer kleinen Datenmenge könnten Sie einfach eine Datei in einem temporären Verzeichnis erstellen und der Drag & Drop-Zwischenablage ein reguläres Dateilisten-Array hinzufügen. Wenn die Datenmenge jedoch groß ist oder wenn die Computerressourcen durch das Generieren der Daten stark beansprucht werden, ist ein langer synchroner Prozess erforderlich. Lange synchrone Prozesse können die Aktualisierung der Benutzeroberfläche deutlich verzögern, sodass der Eindruck entsteht, dass die Anwendung nicht reagiert. Um dieses Problem zu vermeiden, können Sie einen asynchronen Datenprovider erstellen, der von einem Timer gesteuert wird.

Verwenden eines asynchronen Datenprovider in einer Dateizusage

Adobe AIR 2 und höher

Bei Verwendung eines asynchronen Datenprovider-Objekts muss die `isAsync`-Eigenschaft von IFilePromise auf `true` eingestellt sein und das Objekt, das von der `open()`-Methode zurückgegeben wird, muss die IEventDispatcher-Schnittstelle implementieren. Die Laufzeit wartet auf mehrere alternative Ereignisse, sodass verschiedene integrierte Objekte als Datenprovider verwendet werden können. Beispielsweise werden `progress`-Ereignisse von FileStream- und URLStream-Objekten ausgelöst, während `socketData`-Ereignisse von Socket-Objekten ausgelöst werden. Die Laufzeit wartet auf die entsprechenden Ereignisse von allen diesen Objekten.

Die folgenden Ereignisse steuern den Prozess zum Lesen der Daten aus dem Datenprovider-Objekt:

- Event.OPEN – Informiert die Laufzeit, dass die Datenquelle bereit ist.
- ProgressEvent.PROGRESS – Informiert die Laufzeit, dass Daten verfügbar sind. Die Laufzeit liest die verfügbare Datenmenge aus dem Datenprovider-Objekt.
- ProgressEvent.SOCKET_DATA – Informiert die Laufzeit, dass Daten verfügbar sind. Das `socketData`-Ereignis wird von socketbasierten Objekten ausgelöst. Für andere Objekttypen sollte ein `progress`-Ereignis ausgelöst werden. (Die Laufzeit wartet auf beide Ereignisse, um festzustellen, wann Daten gelesen werden können.)
- Event.COMPLETE – Informiert die Laufzeit, dass alle Daten gelesen wurden.
- Event.CLOSE – Informiert die Laufzeit, dass alle Daten gelesen wurden. (Zu diesem Zweck wartet die Laufzeit auf `close` und auf `complete`.)
- IOErrorEvent.IOERROR – Informiert die Laufzeit, dass beim Lesen der Daten ein Fehler aufgetreten ist. Die Laufzeit bricht die Erstellung der Datei ab und ruft die `close()`-Methode von IFilePromise auf.
- SecurityErrorEvent.SECURITY_ERROR – Informiert die Laufzeit, dass ein Sicherheitsfehler aufgetreten ist. Die Laufzeit bricht die Erstellung der Datei ab und ruft die `close()`-Methode von IFilePromise auf.
- HTTPStatusEvent.HTTP_STATUS – Wird zusammen mit `httpResponseStatus` von der Laufzeit verwendet, um sicherzustellen, dass die verfügbaren Daten den gewünschten Inhalt darstellen und keine Fehlermeldung (wie beispielsweise einen 404-Seitenfehler). Objekte auf Basis des HTTP-Protokolls sollten dieses Ereignis ausgeben.
- HTTPStatusEvent.HTTP_RESPONSE_STATUS – Wird zusammen mit `httpStatus` von der Laufzeit verwendet, um sicherzustellen, dass die verfügbaren Daten den gewünschten Inhalt darstellen. Objekte auf Basis des HTTP-Protokolls sollten dieses Ereignis ausgeben.

Der Datenprovider sollte diese Ereignisse in der folgenden Reihenfolge ausgeben:

1. `open-Ereignis`
2. `progress-` oder `socketData`-Ereignisse
3. `complete-` oder `close`-Ereignisse

Hinweis: *Die integrierten Objekte, FileStream, Socket und URLStream, setzen die entsprechenden Ereignisse automatisch ab.*

Im folgenden Beispiel wird eine Dateizusage mit einem benutzerdefinierten asynchronen Datenprovider erstellt. Die Datenprovider-Klasse erweitert ByteArray (für die IDataInput-Unterstützung) und implementiert die IEventDispatcher-Schnittstelle. Bei jedem Timer-Ereignis generiert das Objekt einen Datenblock und setzt ein progress-Ereignis ab, um die Laufzeit über die Verfügbarkeit der Daten zu informieren. Wenn ausreichend Daten generiert wurden, setzt das Objekt ein complete-Ereignis ab.

```
package
{
import flash.events.Event;
import flash.events.EventDispatcher;
import flash.events.IEventDispatcher;
import flash.events.ProgressEvent;
import flash.events.TimerEvent;
import flash.utils.ByteArray;
import flash.utils.Timer;

[Event(name="open", type="flash.events.Event.OPEN")]
[Event(name="complete",  type="flash.events.Event.COMPLETE")]
[Event(name="progress", type="flash.events.ProgressEvent")]
[Event(name="ioError", type="flash.events.IOErrorEvent")]
[Event(name="securityError", type="flash.events.SecurityErrorEvent")]
public class AsyncDataProvider extends ByteArray implements IEventDispatcher
{
    private var dispatcher:EventDispatcher = new EventDispatcher();
    public var fileSize:int = 0; //The number of characters in the file
    private const chunkSize:int = 1000; //Amount of data written per event
    private var dispatchDataTimer:Timer = new Timer( 100 );
    private var opened:Boolean = false;

    public function AsyncDataProvider()
    {
        super();
        dispatchDataTimer.addEventListener( TimerEvent.TIMER, generateData );
    }

    public function begin():void{
        dispatchDataTimer.start();
    }

    public function end():void
    {
        dispatchDataTimer.stop();
    }
    private function generateData( event:Event ):void
    {
        if( !opened )
        {
            var open:Event = new Event( Event.OPEN );
            dispatchEvent( open );
            opened = true;
        }
        else if( position + chunkSize < fileSize )
        {
            for( var i:int = 0; i <= chunkSize; i++ )
            {
                writeUTFBytes( 'A' );
```

```
            }
            //Set position back to the start of the new data
            this.position -= chunkSize;
            var progress:ProgressEvent =
                new ProgressEvent( ProgressEvent.PROGRESS, false, false, bytesAvailable,
bytesAvailable + chunkSize);
            dispatchEvent( progress )
        }
        else
        {
            var complete:Event = new Event( Event.COMPLETE );
            dispatchEvent( complete );
        }
    }
     //IEventDispatcher implementation
    public function addEventListener(type:String, listener:Function,
useCapture:Boolean=false, priority:int=0, useWeakReference:Boolean=false):void
    {
        dispatcher.addEventListener( type, listener, useCapture, priority, useWeakReference );
    }

    public function removeEventListener(type:String, listener:Function,
useCapture:Boolean=false):void
    {
        dispatcher.removeEventListener( type, listener, useCapture );
    }

    public function dispatchEvent(event:Event):Boolean
    {
        return dispatcher.dispatchEvent( event );
    }

    public function hasEventListener(type:String):Boolean
    {
        return dispatcher.hasEventListener( type );
    }

    public function willTrigger(type:String):Boolean
    {
        return dispatcher.willTrigger( type );
    }
}
}
```

Hinweis: *Da die AsyncDataProvider-Klasse im Beispiel ByteArray erweitert, kann sie nicht außerdem EventDispatcher erweitern. Zum Implementieren der IEventDispatcher-Schnittstelle verwendet die Klasse ein internes EventDispatcher-Objekt und leitet die IEventDispatcher-Methodenaufrufe an dieses interne Objekt weiter. Sie könnten auch EventDispatcher erweitern und IDataInput implementieren (oder Sie können beide Schnittstellen implementieren).*

Die asynchrone IFilePromise-Implementierung ist mit der synchronen Implementierung fast identisch. Es bestehen die folgenden Hauptunterschiede: `isAsync` gibt `true` zurück und die `open()`-Methode gibt ein asynchrones Datenobjekt zurück:

```
package
{
    import flash.desktop.IFilePromise;
    import flash.events.ErrorEvent;
    import flash.events.EventDispatcher;
    import flash.utils.IDataInput;

    public class AsynchronousFilePromise extends EventDispatcher implements IFilePromise
    {
        private var fileGenerator:AsyncDataProvider;
        private const fileSize:int = 5000; //size of file data
        private var filePath:String = "AsynchronousFile.txt";

        public function get relativePath():String
        {
            return filePath;
        }

        public function get isAsync():Boolean
        {
            return true;
        }

        public function open():IDataInput
        {
            fileGenerator = new AsyncDataProvider();
            fileGenerator.fileSize = fileSize;
            fileGenerator.begin();
            return fileGenerator;
        }

        public function close():void
        {
            fileGenerator.end();
        }

        public function reportError(e:ErrorEvent):void
        {
            trace("Something went wrong: " + e.errorID + " - " + e.type + ", " + e.text );
        }
    }
}
```

Kapitel 36: Arbeiten mit Menüs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit den Klassen in der Kontextmenü-API können Sie das Kontextmenü in webbasierten Flex- und Flash Player-Anwendungen modifizieren.

Verwenden Sie die Klassen in der nativen Menü-API zum Definieren von Anwendungs-, Fenster-, Kontext- und Popupmenüs in Adobe® AIR®.

Grundlagen von Menüs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine kurze Erklärung sowie Codebeispiele zum Erstellen von nativen Menüs in AIR-Anwendungen finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Hinzufügen nativer Menüs zu AIR-Anwendungen (Flex)
- Hinzufügen nativer Menüs zu AIR-Anwendungen (Flash)

Die nativen Menüklassen geben Zugriff auf die nativen Menüfunktionen des Betriebssystems, unter dem die Anwendung ausgeführt wird. Sie können NativeMenu-Objekte für Anwendungsmenüs (unter Mac OS X verfügbar), Fenstermenüs (unter Windows und Linux verfügbar), Kontextmenüs und Popupmenüs verwenden.

Außerhalb von AIR können Sie die Kontextmenü-Klassen verwenden, um das Kontextmenü zu modifizieren, das Flash Player automatisch anzeigt, wenn ein Benutzer mit der rechten Maustaste bzw. bei gedrückter Befehlstaste auf ein Objekt in Ihrer Anwendung klickt. (Automatische Kontextmenüs werden für AIR-Anwendungen nicht angezeigt.)

Menüklassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Es gibt folgende Menüklassen:

Paket	Klassen
flash.display	• NativeMenu • NativeMenuItem
flash.ui	• ContextMenu • ContextMenuItem
flash.events	• Event • ContextMenuEvent

Menütypen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

AIR unterstützt folgende Menütypen:

Kontextmenüs Kontextmenüs werden nach dem Klicken mit der rechten Maustaste bzw. dem Klicken bei gedrückter Befehlstaste auf ein interaktives Objekt in SWF-Inhalt oder ein Dokumentelement in HTML-Inhalt geöffnet.

In der Flash Player-Laufzeitumgebung werden Kontextmenüs automatisch angezeigt. Mit den ContextMenu- und ContextMenuItem-Klassen können Sie dem Menü eigene Befehle hinzufügen. Sie können auch einige, jedoch nicht alle, der integrierten Befehle entfernen.

In der AIR-Laufzeitumgebung können Sie ein Kontextmenü mit der NativeMenu-Klasse oder der ContextMenu-Klasse erstellen. In HTML-Inhalt in AIR können Sie HTML-Elementen mit den WebKit-HTML- und JavaScript-APIs Kontextmenüs hinzufügen.

Anwendungsmenüs (nur AIR) Ein Anwendungsmenü ist ein globales Menü, das für die gesamte Anwendung gilt. Anwendungsmenüs werden unter Mac OS X, nicht jedoch unter Windows oder Linux unterstützt. Unter Mac OS X erstellt das Betriebssystem automatisch ein Anwendungsmenü. Mithilfe der AIR-Menü-API können Sie den Standardmenüs Menüelemente und Untermenüs hinzufügen. Sie können Listener für den Umgang mit vorhandenen Menübefehlen hinzufügen oder vorhandene Elemente entfernen.

Fenstermenüs (nur AIR) Ein Fenstermenü ist mit einem einzelnen Fenster verknüpft und wird unterhalb der Titelleiste angezeigt. Zum Hinzufügen von Menüs zu einem Fenster können Sie ein NativeMenu-Objekt erstellen und der `menu`-Eigenschaft des NativeWindow-Objekts zuweisen. Fenstermenüs werden unter den Betriebssystemen Windows und Linux unterstützt, jedoch nicht unter Mac OS X. Native Fenstermenüs können nur mit Fenstern mit System-Fensterdesign eingesetzt werden.

Menüs für Dock- und Taskleistensymbole (nur AIR) Diese Symbolmenüs ähneln den Kontextmenüs und werden einem Anwendungssymbol im Dock-Bereich von Mac OS X bzw. im Infobereich der Taskleiste von Windows und Linux zugewiesen. Menüs für Dock- und Taskleistensymbole verwenden die NativeMenu-Klasse. Unter Mac OS X werden die Menüelemente oberhalb der Standardbetriebssystemelemente hinzugefügt. Unter Windows oder Linux gibt es kein Standardmenü.

Popupmenüs (nur AIR) Ein AIR-Popupmenü ist wie ein Kontextmenü, doch ist es nicht unbedingt mit einem bestimmten Anwendungsobjekt oder einer bestimmten Anwendungskomponente verknüpft. Popupmenüs können durch Aufrufen der `display()`-Methode eines beliebigen NativeMenu-Objekts an einer beliebigen Position im Fenster angezeigt werden.

Benutzerdefinierte Menüs Native Menüs werden komplett vom Betriebssystem gezeichnet und existieren daher außerhalb der Flash- und HTML-Renderingmodelle. Anstatt native Menüs zu verwenden, können Sie mit MXML, ActionScript oder JavaScript auch eigene, benutzerdefinierte nicht native Menüs erstellen (in AIR). Derartige Menüs müssen vollständig innerhalb des Anwendungsinhalts gerendert werden.

Flex-Menüs Die Adobe® Flex™-Architekur stellt eine Gruppe von Flex-Menükomponenten bereit. Die Flex-Menüs werden von der Laufzeitumgebung und nicht vom Betriebssystem erstellt, d. h., sie sind keine *nativen* Menüs. Eine Flex-Menükomponente kann für Flex-Fenster ohne System-Fensterdesign verwendet werden. Die Verwendung einer Flex-Menükomponente hat den weiteren Vorteil, dass Sie Menüs deklarativ im MXML-Format festlegen können. Wenn Sie mit der Flex-Architektur arbeiten, verwenden Sie für Fenstermenüs statt der nativen Klassen die Flex-Menüklassen.

Standardmenüs (nur AIR)

Die folgenden Standardmenüs werden vom Betriebssystem oder einer integrierten AIR-Klasse bereitgestellt:

- Anwendungsmenü unter Mac OS X

- Dock-Symbolmenü unter Mac OS X
- Kontextmenü für ausgewählten Text und ausgewählte Bilder in HTML-Inhalt
- Kontextmenü für ausgewählten Text in einem TextField-Objekt (bzw. einem Objekt, das das TextField erweitert)

Kontextmenüs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei SWF-Inhalten können Sie jedes Objekt, das vom InteractiveObject erbt, durch Zuweisen eines Menüobjekts zu der entsprechenden `contextMenu`-Eigenschaft mit einem Kontextmenü versehen. Standardmäßig sind mehrere Befehle enthalten, u. a. „Vorwärts“, „Zurück“, „Drucken“, „Qualität“ und „Zoom“. In der AIR-Laufzeitumgebung kann das `contextMenu` zugewiesene Menüobjekt vom Typ NativeMenu oder ContextMenu sein. In der Flash Player-Laufzeitumgebung steht nur die ContextMenu-Klasse zur Verfügung.

Sie können auf native Menüereignisse oder Kontextmenüereignisse warten, wenn Sie die ContextMenu- und ContextMenuItem-Klassen verwenden; beide werden ausgelöst. Ein Vorteil der vom ContextMenuEvent-Objekt bereitgestellten Eigenschaften liegt darin, dass contextMenuOwner das Objekt identifiziert, dem das Menü zugeordnet ist, und dass `mouseTarget` das Objekt identifiziert, auf das zum Öffnen des Menüs geklickt wurde. Diese Informationen sind vom NativeMenuEvent-Objekt nicht verfügbar.

Das folgende Beispiel erstellt ein Sprite-Objekt und fügt ein einfaches Kontextmenü mit Bearbeitungsbefehlen hinzu:

```
 var sprite:Sprite = new Sprite();
sprite.contextMenu = createContextMenu()
private function createContextMenu():ContextMenu{
    var editContextMenu:ContextMenu = new ContextMenu();
    var cutItem:ContextMenuItem = new ContextMenuItem("Cut")
    cutItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, doCutCommand);
    editContextMenu.customItems.push(cutItem);

    var copyItem:ContextMenuItem = new ContextMenuItem("Copy")
    copyItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, doCopyCommand);
    editContextMenu.customItems.push(copyItem);

    var pasteItem:ContextMenuItem = new ContextMenuItem("Paste")
    pasteItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, doPasteCommand);
    editContextMenu.customItems.push(pasteItem);

    return editContextMenu
}
private function doCutCommand(event:ContextMenuEvent):void{trace("cut");}
private function doCopyCommand(event:ContextMenuEvent):void{trace("copy");}
private function doPasteCommand(event:ContextMenuEvent):void{trace("paste");}
```

Hinweis: *Im Gegensatz zu SWF-Inhalt, der in einer Browserumgebung angezeigt wird, enthalten Kontextmenüs in AIR keine integrierten Befehle.*

Anpassen von Flash Player-Kontextmenüs

In einem Browser oder Projektor haben Kontextmenüs in SWF-Inhalten immer integrierte Menüpunkte. Sie können alle dieser Standardbefehle mit Ausnahme der Optionen „Einstellungen“ und „Info“ aus dem Menü entfernen. Durch Setzen der Stage-Eigenschaft `showDefaultContextMenu` auf `false` werden diese Befehle aus dem Kontextmenü entfernt.

Wenn Sie für ein bestimmtes Anzeigeobjekt ein benutzerdefiniertes Kontextmenü erstellen möchten, müssen Sie eine neue Instanz der ContextMenu-Klasse erstellen, die `hideBuiltInItems()`-Methode aufrufen und dann die neue Instanz der `contextMenu`-Eigenschaft der entsprechenden DisplayObject-Instanz zuweisen. Im folgenden Beispiel wird ein dynamisch gezeichnetes Rechteck mit einem Kontextmenübefehl erstellt, der dessen Farbe auf einen zufällig ausgewählten Wert setzt:

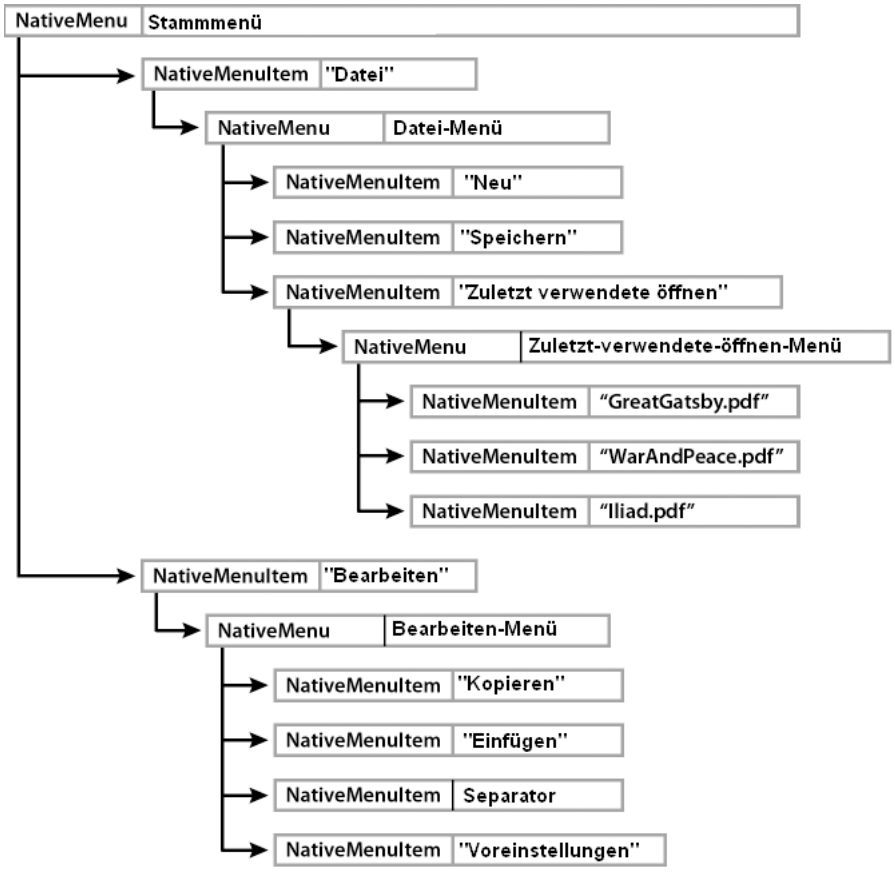
```
var square:Sprite = new Sprite();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.x =
square.y = 10;
addChild(square);

var menuItem:ContextMenuItem = new ContextMenuItem("Change Color");
menuItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT,changeColor);
var customContextMenu:ContextMenu = new ContextMenu();
customContextMenu.hideBuiltInItems();
customContextMenu.customItems.push(menuItem);
square.contextMenu = customContextMenu;

function changeColor(event:ContextMenuEvent):void
{
    square.transform.colorTransform = getRandomColor();
}
function getRandomColor():ColorTransform
{
    return new ColorTransform(Math.random(), Math.random(), Math.random(),1,(Math.random() *
512) - 255, (Math.random() * 512) -255, (Math.random() * 512) - 255, 0);
}
```

Struktur von nativen Menüs (AIR)

Adobe AIR 1.0 und höher

Native Menüs sind von Natur aus hierarchisch. NativeMenu-Objekte können untergeordnete NativeMenuItem-Objekte enthalten. NativeMenuItem-Objekte, die Untermenüs repräsentieren, können wiederum NativeMenu-Objekte enthalten. Das Menüobjekt, das sich in der Struktur auf der obersten Ebene oder der Stammebene befindet, stellt die Menüleiste von Anwendungs- und Fenstermenüs dar. (Kontext-, Symbol- und Popupmenüs haben keine Menüleiste.)

Das folgende Diagramm veranschaulicht die Struktur eines typischen Menüs. Das Stammmenü repräsentiert die Menüleiste und enthält zwei Menüelemente, die auf die Untermenüs *Datei* und *Bearbeiten* verweisen. Das Untermenü „Datei“ in dieser Struktur enthält zwei Befehlselemente und ein Element, das auf das Untermenü *Zuletzt geöffnete Dateien* verweist, welches wiederum drei Elemente enthält. Das Untermenü „Bearbeiten“ enthält drei Befehle und eine Trennlinie.



Zum Definieren eines Untermenüs sind sowohl ein NativeMenu- als auch ein NativeMenuItem-Objekt erforderlich. Das NativeMenuItem-Objekt definiert die im übergeordneten Menü angezeigte Bezeichnung und ermöglicht dem Benutzer das Öffnen des Untermenüs. Das NativeMenu-Objekt dient als Container für Elemente im Untermenü. Das NativeMenuItem-Objekt verweist über die `submenu`-Eigenschaft des NativeMenuItem auf das NativeMenu-Objekt.

Ein Codebeispiel, das dieses Menü erstellt, finden Sie unter „Beispiel für natives Menü: Fenster- und Anwendungsmenü (AIR)“ auf Seite 685.

Menüereignisse

Adobe AIR 1.0 und höher

NativeMenu- und NativeMenuItem-Objekte lösen beide `preparing`-, `displaying`- und `select`-Ereignisse aus:

preparing: Wenn das Objekt dabei ist, eine Benutzerinteraktion zu beginnen, lösen das Menü und seine Menüelemente ein `preparing`-Ereignis an registrierte Listener aus. Interaktionen sind beispielsweise das Öffnen des Menüs oder die Auswahl eines Menüelements über einen Tastaturbefehl.

Hinweis: *Das `preparing`-Ereignis ist nur für Adobe AIR 2.6 und höher verfügbar.*

displaying: Unmittelbar vor der Anzeige eines Menüs lösen das Menü und seine Menüelemente ein `displaying`-Ereignis für alle registrierten Listener aus.

Die `preparing`- und `displaying`-Ereignisse bieten Ihnen die Möglichkeit, die Menüinhalte oder das Erscheinungsbild der Menüelemente zu aktualisieren, bevor diese dem Benutzer präsentiert werden. Sie könnten beispielsweise im Listener für das `displaying`-Ereignis eines Menüs für „Zuletzt geöffnete Dateien" die Menüelemente so ändern, dass eine Liste der zuletzt angezeigten Dokumente zu sehen ist.

Wenn Sie das Menüelement entfernen, dessen Tastaturbefehl ein `preparing`-Ereignis ausgelöst hat, wird die Menüinteraktion abgebrochen. In diesem Fall wird kein `select`-Ereignis ausgelöst.

Die `target`- und `currentTarget`-Eigenschaften des Ereignisses sind beide das Objekt, für das der Listener registriert ist: entweder das Menü selbst oder eines seiner Elemente.

Das `preparing`-Ereignis wird vor dem `displaying`-Ereignis ausgelöst. Normalerweise wird auf eines der beiden Ereignisse gewartet, aber nicht auf beide.

Auswahl: Wenn der Benutzer ein Befehlselement auswählt, löst dieses für alle registrierten Listener ein `select`-Ereignis aus. Untermenü- und Trennelemente können nicht ausgewählt werden und lösen daher nie ein `select`-Ereignis aus.

Ein `select`-Ereignis beginnt bei einem Menüelement und setzt sich über das Menü, in dem das Element enthalten ist, bis zum Stammmenü fort. Sie können `select`-Ereignisse direkt auf Elementebene oder weiter oben in der Menüstruktur überwachen. Wenn Sie das `select`-Ereignis auf Menüebene überwachen, können Sie das ausgewählte Element mit der `target`-Eigenschaft des Ereignisses identifizieren. Während der Fortsetzung des Ereignisses durch die Menühierarchie identifiziert die `currentTarget`-Eigenschaft des Ereignisobjekts das aktuelle Menüobjekt.

Hinweis: *ContextMenu- und ContextMenuItem-Objekte lösen die Ereignisse `menuItemSelect` und `menuSelect` sowie `select`, `preparing` und `displaying` aus.*

Zugriffstasten für native Menübefehle (AIR)

Adobe AIR 1.0 und höher

Sie können einem Menübefehl Zugriffstasten zuweisen. Das Menüelement löst für alle registrierten Listener beim Drücken der Taste bzw. Tastenkombination ein `select`-Ereignis aus. Das Menü, das das Element enthält, muss zum Menü der Anwendung oder zum aktiven Fenster gehören, damit der Befehl aufgerufen wird.

Zugriffstasten bestehen aus zwei Teilen: einem String, der der Haupttaste entspricht, und einem Array von Zusatztasten, die ebenfalls gedrückt werden müssen. Zum Zuweisen der Haupttaste legen Sie für die `keyEquivalent`-Eigenschaft des Menüelements ein einzelnes Zeichen für diese Taste fest. Wenn Sie einen Großbuchstaben verwenden, wird die Umschalttaste dem Zusatztastenarray automatisch hinzugefügt.

Unter Mac OS X wird als Standardzusatztaste die Befehlstaste (`Keyboard.COMMAND`) verwendet. Unter Windows und Linux ist dies die Strg-Taste (`Keyboard.CONTROL`). Diese Standardtasten werden dem Zusatztastenarray automatisch hinzugefügt. Wenn andere Zusatztasten verwendet werden sollen, weisen Sie der `keyEquivalentModifiers`-Eigenschaft ein neues Array zu, das die gewünschten Tastencodes enthält. Das Standardarray wird überschrieben. Die Umschalttaste wird automatisch hinzugefügt, wenn der der `keyEquivalent`-Eigenschaft zugewiesene String ein Großbuchstabe ist, ganz gleich, ob Sie die Standardzusatztaten verwenden oder ein eigenes Zusatztastenarray zuweisen. Konstanten für die für Zusatztasten zu verwendenden Tastencodes sind in der Keyboard-Klasse definiert.

Der zugewiesene String für Zugriffstasten wird automatisch neben dem Namen des Menüelements angezeigt. Das Format hängt vom Betriebssystem des Benutzers und den Systemeinstellungen ab.

Hinweis: *Wenn Sie einem Zusatztastenarray unter Windows den* `Keyboard.COMMAND`*-Wert zuweisen, werden im Menü keine Zugriffstasten angezeigt. Zum Aktivieren des Menübefehls muss die Strg-Taste verwendet werden.*

Im folgenden Beispiel werden einem Menüelement die Zugriffstasten `Strg+Umschalt+G` zugewiesen:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
```

Dieses Beispiel legt das Zusatztastenarray direkt fest und weist so `Strg+Umschalt+G` als Zugriffstasten zu:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
item.keyEquivalentModifiers = [Keyboard.CONTROL];
```

Hinweis: *Zugriffstasten werden nur für Anwendungs- und Fenstermenüs ausgelöst. Wenn Sie einem Kontext- oder Popupmenü Zugriffstasten hinzufügen, werden diese in der Menübezeichnung angezeigt, der verknüpfte Menübefehl aber nie aufgerufen.*

Mnemonische Zeichen (AIR)

Adobe AIR 1.0 und höher

Mnemonische Zeichen sind ein Teil der Tastaturschnittstelle des Betriebssystems für Menüs. Sowohl unter Mac OS X als auch unter Windows und Linux können Benutzer zum Öffnen von Menüs und Auswählen von Befehlen die Tastatur verwenden. Allerdings gibt es hier geringfügige Unterschiede.

Unter Mac OS X gibt der Benutzer die ersten ein oder zwei Buchstaben des Menüs oder Befehls ein und drückt dann die Return-Taste. Die `mnemonicIndex`-Eigenschaft wird ignoriert.

Unter Windows ist nur ein einzelner Buchstabe relevant. Standardmäßig ist der relevante Buchstabe das erste Zeichen der Bezeichnung. Wenn Sie einem Menüelement jedoch ein mnemonisches Zeichen zuweisen, entspricht dies dem relevanten Buchstaben. Wenn zwei Elemente in einem Menü (unabhängig davon, ob ein mnemoisches Zeichen zugewiesen wurde) das gleiche relevante Zeichen aufweisen, ändert sich die Tastaturinteraktion des Benutzers mit dem Menü geringfügig. Statt des Einzelbuchstabens muss der Benutzer zum Auswählen des Menüs oder Befehls diesen Buchstaben nun so oft drücken, bis das gewünschte Element hervorgehoben wird, und anschließend zum Durchführen der Auswahl die Eingabetaste betätigen. Um ein einheitliches Verhalten zu gewährleisten, sollten Sie bei Fenstermenüs jedem Menüelement ein eindeutiges mnemonisches Zeichen zuweisen.

Unter Linux wird kein standardmäßiges mnemonisches Zeichen bereitgestellt. Sie müssen einen Wert für die `mnemonicIndex`-Eigenschaft eines Menüelements angeben, um ein mnemonisches Zeichen festzulegen.

Geben Sie das mnemonische Zeichen als Index im Bezeichnungsstring an. Der Index des ersten Zeichens in einer Bezeichnung ist 0. Wenn Sie für ein Menüelement namens „Format“ als mnemonisches Zeichen „r“ verwenden möchten, müssen Sie für die `mnemonicIndex`-Eigenschaft 2 festlegen.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.mnemonicIndex = 2;
```

Menüelementstatus

Adobe AIR 1.0 und höher

Menüelemente haben zwei Statuseigenschaften, `checked` und `enabled`:

checked Setzen Sie diese Einstellung auf `true`, um neben der Elementbezeichnung ein Häkchen anzuzeigen.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.checked = true;
```

enabled Schalten Sie den Wert zwischen `true` und `false` um, um zu steuern, ob der Befehl aktiviert wurde oder nicht. Deaktivierte Elemente sind grau dargestellt und lösen keine `select`-Ereignisse aus.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.enabled = false;
```

Anheften von Objekten an Menüelemente

Adobe AIR 1.0 und höher

Mit der `data`-Eigenschaft der NativeMenuItem-Klasse können Sie in jedem Element auf ein willkürliches Objekt verweisen. In einem Menü der zuletzt geöffneten Dateien könnten Sie z. B. jedem Menüelement das File-Objekt für jedes Dokument zuweisen.

```
var file:File = File.applicationStorageDirectory.resolvePath("GreatGatsby.pdf")
var menuItem:NativeMenuItem = docMenu.addItem(new NativeMenuItem(file.name));
menuItem.data = file;
```

Erstellen nativer Menüs (AIR)

Adobe AIR 1.0 und höher

In diesem Thema erfahren Sie, wie die verschiedenen von AIR unterstützten nativen Menüs erstellt werden.

Erstellen von Stammmenüobjekten

Adobe AIR 1.0 und höher

Verwenden Sie den NativeMenu-Konstruktor zum Erstellen eines NativeMenu-Objekts, das als Stamm des Menüs dienen soll:

```
var root:NativeMenu = new NativeMenu();
```

Bei Anwendungs- und Fenstermenüs entspricht die Menüleiste dem Stammmenü und sollte nur Elemente enthalten, die Untermenüs öffnen. Kontext- und Popupmenüs haben keine Menüleiste. Daher kann das Stammmenü Befehle, Trennlinien und Untermenüs enthalten.

Nachdem Sie das Menü erstellt haben, können Sie ihm Elemente hinzufügen. Elemente werden in der Reihenfolge im Menü angezeigt, in der sie hinzugefügt wurden, sofern Sie die Elmente nicht mit der `addItemAt()`-Methode eines Menüobjekts an einem bestimmten Index hinzufügen.

Legen Sie das Menü als Anwendungs-, Fenster-, Symbol- oder Kontextmenü fest oder zeigen Sie es als Popupmenü an, wie in den nachstehenden Abschnitten veranschaulicht:

Festlegen von Anwendungsmenüs oder Fenstermenüs

Ihr Code muss sowohl Anwendungsmenüs (unter Mac OS unterstützt) als auch Fenstermenüs (unter anderen Betriebssystemen unterstützt) verarbeiten können.

```
var root:NativeMenu = new NativeMenu();
if (NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu = root;
}
else if (NativeWindow.supportsMenu)
{
    nativeWindow.menu = root;
}
```

Hinweis: *Unter Mac OS enthält ein Menü laut Definition Standardelemente für jede Anwendung. Durch Zuweisen eines NativeMenu-Objekts zur* `menu`*-Eigenschaft des NativeApplication-Objekts wird das Standardmenü ersetzt. Sie müssen das Standardmenü jedoch nicht ersetzen, sondern können dieses direkt verwenden.*

Adobe Flex enthält die FlexNativeMenu-Klasse, mit der ganz einfach Menüs erstellt werden können, die auf verschiedenen Plattformen funktionieren. Wenn Sie die Flex-Architektur einsetzen, verwenden Sie die FlexNativeMenu-Klassen anstelle der NativeMenu-Klasse.

Festlegen von Kontextmenüs für interaktive Objekte

```
interactiveObject.contextMenu = root;
```

Festlegen von Menüs für Dock-Symbole oder von Menüs für Infobereich-Symbole

Ihr Code muss sowohl Anwendungsmenüs (unter Mac OS unterstützt) als auch Fenstermenüs (unter anderen Betriebssystemen unterstützt) verarbeiten können.

```
if (NativeApplication.supportsSystemTrayIcon)
{
    SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
}
else if (NativeApplication.supportsDockIcon)
{
    DockIcon(NativeApplication.nativeApplication.icon).menu = root;
}
```

Hinweis: *Das Mac OS X definiert ein Standardmenü für das Dock-Symbol der Anwendung. Wenn Sie der menu-Eigenschaft des DockIcon-Objekts ein neues NativeMenu zuweisen, werden die Elemente in diesem Menü oberhalb der Standardelemente angezeigt. Es ist nicht möglich, Standardmenüelemente zu entfernen, zu ändern oder auf diese zuzugreifen.*

Anzeigen von Menüs als Popupmenüs

```
root.display(stage, x, y);
```

Verwandte Hilfethemen

Entwickeln von plattformübergreifenden AIR-Anwendungen

Erstellen von Untermenüs

Adobe AIR 1.0 und höher

Zum Erstellen eines Untermenüs fügen Sie dem übergeordneten Menü ein NativeMenuItem-Objekt hinzu und weisen dann das NativeMenu-Objekt, das das Untermenü definiert, der `submenu`-Eigenschaft des Elements zu. AIR bietet zwei Möglichkeiten zum Erstellen von Untermenüelementen und dem zugehörigen Menüobjekt:

Mit der `addSubmenu()`-Methode können Sie ein Menüelement und das zugehörige Menüobjekt in einem Schritt erstellen:

```
var editMenuItem:NativeMenuItem = root.addSubmenu(new NativeMenu(), "Edit");
```

Sie können zum Erstellen des Menüelements und Zuweisen des Menüobjekts zur `submenu`-Eigenschaft aber auch separate Schritte ausführen:

```
var editMenuItem:NativeMenuItem = root.addItem("Edit", false);
editMenuItem.submenu = new NativeMenu();
```

Erstellen von Menübefehlen

Adobe AIR 1.0 und höher

Fügen Sie zum Erstellen eines Menübefehls einem Menü ein NativeMenuItem-Objekt und einen Listener hinzu, der auf die Funktion verweist, die den Menübefehl implementiert:

```
var copy:NativeMenuItem = new NativeMenuItem("Copy", false);
copy.addEventListener(Event.SELECT, onCopyCommand);
editMenu.addItem(copy);
```

Sie können das `select`-Ereignis (wie im Beispiel) im Befehl selbst oder im übergeordneten Menüobjekt überwachen.

Hinweis: *Menüelemente, die Untermenüs und Trennlinien repräsentieren, lösen keine `select`-Ereignisse aus und können daher nicht als Befehle verwendet werden.*

Erstellen von Menütrennlinien

Adobe AIR 1.0 und höher

Erstellen Sie zum Erstellen einer Trennlinie ein NativeMenuItem und setzen Sie den `isSeparator`-Parameter im Konstruktor auf `true`. Fügen Sie das Trennelement dann an der richtigen Stelle im Menü hinzu:

```
var separatorA:NativeMenuItem = new NativeMenuItem("A", true);
editMenu.addItem(separatorA);
```

Die gegebenenfalls für die Trennlinie angegebene Bezeichnung wird nicht angezeigt.

Kontextmenüs in HTML (AIR)

Adobe AIR 1.0 und höher

In HTML-Inhalt, der mithilfe des HTMLLoader-Objekts angezeigt wird, können Sie ein Kontextmenü mit dem `contextmenu`-Ereignis anzeigen. Standardmäßig wird ein Kontextmenü automatisch angezeigt, wenn der Benutzer für ausgewählten Text (durch Klicken mit der rechten Maustaste bzw. der Befehlstaste) ein Kontextmenüereignis aufruft. Um ein Öffnen des Standardmenüs zu verhindern, überwachen Sie das `contextmenu`-Ereignis und rufen Sie die `preventDefault()`-Methode des Ereignisobjekts auf:

```
function showContextMenu(event){
    event.preventDefault();
}
```

Sie können dann mit den DHTML-Methoden oder durch Anzeigen eines nativen AIR-Kontextmenüs ein benutzerdefiniertes Kontextmenü einblenden. Das folgende Beispiel reagiert auf das HTML-`contextmenu`-Ereignis durch Aufrufen der `display()`-Methode des Menüs zum Anzeigen eines nativen Kontextmenüs:

```
<html>
<head>
<script src="AIRAliases.js" language="JavaScript" type="text/javascript"></script>
<script language="javascript" type="text/javascript">

function showContextMenu(event){
    event.preventDefault();
    contextMenu.display(window.nativeWindow.stage, event.clientX, event.clientY);
}

function createContextMenu(){
    var menu = new air.NativeMenu();
    var command = menu.addItem(new air.NativeMenuItem("Custom command"));
    command.addEventListener(air.Event.SELECT, onCommand);
    return menu;
}

function onCommand(){
    air.trace("Context command invoked.");
}

var contextMenu = createContextMenu();
</script>
</head>
<body>
<p oncontextmenu="showContextMenu(event)" style="-khtml-user-select:auto;">Custom context
menu.</p>
</body>
</html>
```

Anzeigen von nativen Popupmenüs (AIR)

Adobe AIR 1.0 und höher

Durch Aufrufen der `display()`-Methode des Menüs können Sie beliebige NativeMenu-Objekte zu einem beliebigen Zeitpunkt und an einer beliebigen Stelle oberhalb des Fensters anzeigen. Für diese Methode ist ein Verweis auf die Phase erforderlich. Daher kann nur Inhalt in der Anwendungs-Sandbox ein Menü als Popupmenü anzeigen.

Die folgende Methode zeigt beim Klicken mit der Maus das von einem NativeMenu-Objekt mit Namen `popupMenu` definierte Menü an:

```
private function onMouseClick(event:MouseEvent):void {
    popupMenu.display(event.target.stage, event.stageX, event.stageY);
}
```

Hinweis: *Das Menü muss nicht als direkte Reaktion auf ein Ereignis angezeigt werden. Die `display()`-Funktion kann von einer beliebigen Methode aufgerufen werden.*

Umgang mit Menüereignissen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Menü löst Ereignisse aus, wenn der Benutzer das Menü oder ein Menüelement auswählt.

Ereigniszusammenfassung für Menüklassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Fügen Sie Menüs oder einzelnen Elementen Listener für den Umgang mit Menüelementen hinzu.

Object	Ausgelöste Ereignisse
NativeMenu (AIR)	Event.PREPARING (Adobe AIR 2.6 und höher) Event.DISPLAYING Event.SELECT (aus untergeordneten Elementen und Untermenüs übertragen)
NativeMenuItem (AIR)	Event.PREPARING (Adobe AIR 2.6 und höher) Event.SELECT Event.DISPLAYING (aus dem übergeordneten Menü übertragen)
ContextMenu	ContextMenuEvent.MENU_SELECT
ContextMenuItem	ContextMenuEvent.MENU_ITEM_SELECT Event.SELECT (AIR)

Auswählen von Menüereignissen

Adobe AIR 1.0 und höher

Fügen Sie dem NativeMenuItem einen Listener für das `select`-Ereignis hinzu, um einen Klick auf ein Menüelement zu verarbeiten:

```
var menuCommandX:NativeMenuItem = new NativeMenuItem("Command X");
menuCommandX.addEventListener(Event.SELECT, doCommandX)
```

Da sich `select`-Ereignisse nach oben durch die Menüs, in denen sie enthalten sind, fortsetzen, können Sie auch in einem übergeordneten Menü einen Listener für diese Ereignisse verwenden. Beim Überwachen auf der Menüebene können Sie mithilfe der `target`-Eigenschaft des Ereignisobjekts ermitteln, welcher Menübefehl ausgewählt wurde. Im folgenden Beispiel wird die Bezeichnung des ausgewählten Befehls verfolgt:

```
 var colorMenuItem:NativeMenuItem = new NativeMenuItem("Choose a color");
var colorMenu:NativeMenu = new NativeMenu();
colorMenuItem.submenu = colorMenu;

var red:NativeMenuItem = new NativeMenuItem("Red");
var green:NativeMenuItem = new NativeMenuItem("Green");
var blue:NativeMenuItem = new NativeMenuItem("Blue");
colorMenu.addItem(red);
colorMenu.addItem(green);
colorMenu.addItem(blue);

if(NativeApplication.supportsMenu){
    NativeApplication.nativeApplication.menu.addItem(colorMenuItem);
    NativeApplication.nativeApplication.menu.addEventListener(Event.SELECT, colorChoice);
} else if (NativeWindow.supportsMenu){
    var windowMenu:NativeMenu = new NativeMenu();
    this.stage.nativeWindow.menu = windowMenu;
    windowMenu.addItem(colorMenuItem);
    windowMenu.addEventListener(Event.SELECT, colorChoice);
}

function colorChoice(event:Event):void {
    var menuItem:NativeMenuItem = event.target as NativeMenuItem;
    trace(menuItem.label + " has been selected");
}
```

Wenn Sie die ContextMenuItem-Klasse verwenden, können Sie entweder das `select`- oder das `menuItemSelect`-Ereignis überwachen. Das `menuItemSelect`-Ereignis liefert zusätzliche Informationen zum Objekt, zu dem das Kontextmenü gehört, wird aber nicht nach oben durch die Menüs, in denen es enthalten ist, fortgesetzt.

Anzeigen von Menüereignissen

Adobe AIR 1.0 und höher

Sie können für das `displaying`-Ereignis, das ausgelöst wird, bevor ein Menü angezeigt wird, einen Listener hinzufügen, der das Öffnen des Menüs übernimmt. Mit dem displaying-Ereignis können Sie das Menü aktualisieren, indem Sie beispielsweise Elemente hinzufügen oder entfernen oder den enabled- oder checked-Status der individuellen Elemente aktualisieren. Sie können auch das `menuSelect`-Ereignis von einem ContextMenu-Objekt überwachen.

In AIR 2.6 und höher können Sie das `preparing`-Ereignis verwenden, um ein Menü zu aktualisieren, nachdem der Benutzer ein Menü angezeigt oder ein Menüelement über einen Tastaturbefehl ausgewählt hat.

Beispiel für natives Menü: Fenster- und Anwendungsmenü (AIR)

Adobe AIR 1.0 und höher

Im folgenden Beispiel wird das unter „Struktur von nativen Menüs (AIR)“ auf Seite 676 abgebildete Menü erstellt.

Das Menü ist so konzipiert, dass es sowohl unter Windows (nur Fenstermenüs) als auch unter Mac OS X (nur Anwendungsmenüs) funktioniert. Um zwischen den beiden Betriebssystemen zu unterscheiden, prüft der MenuExample-Klassenkonstruktor die statischen `supportsMenu`-Eigenschaften der NativeWindow- und NativeApplication-Klasse. Wenn `NativeWindow.supportsMenutrue` ist, erstellt der Konstruktor zunächst ein NativeMenu-Objekt für das Fenster und anschließend die Untermenüs „Datei“ und „Bearbeiten“ und fügt diese hinzu. Wenn `NativeApplication.supportsMenutrue` ist, erstellt der Konstruktor die Menüs „Datei“ und „Bearbeiten“ im bestehenden, vom Mac OS X-Betriebssystem bereitgestellten Menü und fügt diese hinzu.

In diesem Beispiel wird auch der Umgang mit Menüereignissen veranschaulicht. Das `select`-Ereignis wird auf Element- und Menüebene verarbeitet. Jedes Menü in der Kette, vom Menü mit dem ausgewählten Element bis hin zum Stammelement, reagiert auf das `select`-Ereignis. Das `displaying`-Ereignis wird mit dem Menü „Zuletzt geöffnete Dateien“ verwendet. Unmittelbar, bevor das Menü geöffnet wird, werden die enthaltenen Elemente anhand des Arrays zuletzt geöffneter Dokumente aktualisiert (in diesem Beispiel bleibt diese Liste allerdings unverändert). Sie können auch individuelle Elemente auf `displaying`-Ereignisse hin überwachen; dies wird in diesem Beispiel jedoch nicht veranschaulicht.

```
package {
    import flash.display.NativeMenu;
    import flash.display.NativeMenuItem;
    import flash.display.NativeWindow;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.filesystem.File;
    import flash.desktop.NativeApplication;

    public class MenuExample extends Sprite
    {
        private var recentDocuments:Array =
            new Array(new File("app-storage:/GreatGatsby.pdf"),
                      new File("app-storage:/WarAndPeace.pdf"),
                      new File("app-storage:/Iliad.pdf"));

        public function MenuExample()
        {
            var fileMenu:NativeMenuItem;
            var editMenu:NativeMenuItem;

            if (NativeWindow.supportsMenu){
                stage.nativeWindow.menu = new NativeMenu();
                stage.nativeWindow.menu.addEventListener(Event.SELECT, selectCommandMenu);
                fileMenu = stage.nativeWindow.menu.addItem(new NativeMenuItem("File"));
                fileMenu.submenu = createFileMenu();
                editMenu = stage.nativeWindow.menu.addItem(new NativeMenuItem("Edit"));
                editMenu.submenu = createEditMenu();
            }

            if (NativeApplication.supportsMenu){
                NativeApplication.nativeApplication.menu.addEventListener(Event.SELECT,
selectCommandMenu);
                fileMenu = NativeApplication.nativeApplication.menu.addItem(new
NativeMenuItem("File"));
                fileMenu.submenu = createFileMenu();
                editMenu = NativeApplication.nativeApplication.menu.addItem(new
NativeMenuItem("Edit"));
                editMenu.submenu = createEditMenu();
            }
```

```
    }

    public function createFileMenu():NativeMenu {
        var fileMenu:NativeMenu = new NativeMenu();
        fileMenu.addEventListener(Event.SELECT, selectCommandMenu);

        var newCommand:NativeMenuItem = fileMenu.addItem(new NativeMenuItem("New"));
        newCommand.addEventListener(Event.SELECT, selectCommand);
        var saveCommand:NativeMenuItem = fileMenu.addItem(new NativeMenuItem("Save"));
        saveCommand.addEventListener(Event.SELECT, selectCommand);
        var openRecentMenu:NativeMenuItem =
                fileMenu.addItem(new NativeMenuItem("Open Recent"));
        openRecentMenu.submenu = new NativeMenu();
        openRecentMenu.submenu.addEventListener(Event.DISPLAYING,
                                        updateRecentDocumentMenu);
        openRecentMenu.submenu.addEventListener(Event.SELECT, selectCommandMenu);

        return fileMenu;
    }

    public function createEditMenu():NativeMenu {
        var editMenu:NativeMenu = new NativeMenu();
        editMenu.addEventListener(Event.SELECT, selectCommandMenu);

        var copyCommand:NativeMenuItem = editMenu.addItem(new NativeMenuItem("Copy"));
        copyCommand.addEventListener(Event.SELECT, selectCommand);
        copyCommand.keyEquivalent = "c";
        var pasteCommand:NativeMenuItem =
                editMenu.addItem(new NativeMenuItem("Paste"));
        pasteCommand.addEventListener(Event.SELECT, selectCommand);
        pasteCommand.keyEquivalent = "v";
        editMenu.addItem(new NativeMenuItem("", true));
        var preferencesCommand:NativeMenuItem =
                editMenu.addItem(new NativeMenuItem("Preferences"));
        preferencesCommand.addEventListener(Event.SELECT, selectCommand);

        return editMenu;
    }

    private function updateRecentDocumentMenu(event:Event):void {
        trace("Updating recent document menu.");
        var docMenu:NativeMenu = NativeMenu(event.target);

        for each (var item:NativeMenuItem in docMenu.items) {
            docMenu.removeItem(item);
        }

        for each (var file:File in recentDocuments) {
            var menuItem:NativeMenuItem =
                    docMenu.addItem(new NativeMenuItem(file.name));
            menuItem.data = file;
            menuItem.addEventListener(Event.SELECT, selectRecentDocument);
        }
    }

    private function selectRecentDocument(event:Event):void {
        trace("Selected recent document: " + event.target.data.name);
```

```
        }

        private function selectCommand(event:Event):void {
            trace("Selected command: " + event.target.label);
        }

        private function selectCommandMenu(event:Event):void {
            if (event.currentTarget.parent != null) {
                var menuItem:NativeMenuItem =
                        findItemForMenu(NativeMenu(event.currentTarget));
                if (menuItem != null) {
                    trace("Select event for \"" +
                            event.target.label +
                            "\" command handled by menu: " +
                            menuItem.label);
                }
            } else {
                trace("Select event for \"" +
                        event.target.label +
                        "\" command handled by root menu.");
            }
        }

        private function findItemForMenu(menu:NativeMenu):NativeMenuItem {
            for each (var item:NativeMenuItem in menu.parent.items) {
                if (item != null) {
                    if (item.submenu == menu) {
                        return item;
                    }
                }
            }
            return null;
        }
    }
}
```

Kapitel 37: Taskleisten-Symbol in AIR

Adobe AIR 1.0 und höher

Viele Betriebssysteme bieten eine Taskleiste, wie etwa das Mac OS X-Dock, in der eine Anwendung durch ein Symbol dargestellt wird. Adobe® AIR® bietet mithilfe der `NativeApplication.nativeApplication.icon`-Eigenschaft eine Schnittstelle zur Interaktion mit Anwendungssymbolen in der Taskleiste.

- Verwenden von Symbolen im Infobereich und Dock (Flex)
- Verwenden von Symbolen im Infobereich und Dock (Flash)

Verwandte Hilfethemen

flash.desktop.NativeApplication

flash.desktop.DockIcon

flash.desktop.SystemTrayIcon

Taskleisten-Symbole

Adobe AIR 1.0 und höher

AIR erstellt automatisch das `NativeApplication.nativeApplication.icon`-Objekt. Je nach Betriebssystem handelt es sich beim Objekttyp entweder um „DockIcon“ oder „SystemTrayIcon“. Mithilfe der `NativeApplication.supportsDockIcon`- und `NativeApplication.supportsSystemTrayIcon`-Eigenschaften können Sie feststellen, welche dieser InteractiveIcon-Unterklassen von AIR auf dem aktuellen Betriebssystem unterstützt wird. Die InteractiveIcon-Basisklasse bietet die Eigenschaften `width`, `height` und `bitmaps`, mit der Sie das für das Symbol verwendete Bild ändern können. Wenn Sie jedoch eine für „DockIcon“ oder „SystemTrayIcon“ spezifische Eigenschaft auf dem falschen Betriebssystem aufrufen, kommt es zu einem Laufzeitfehler.

Um das für ein Symbol verwendete Bild festzulegen oder zu ändern, erstellen Sie ein Array, das ein oder mehrere Bilder enthält, und weisen es der `NativeApplication.nativeApplication.icon.bitmaps`-Eigenschaft zu. Die Größe der Taskleisten-Symbole kann sich von Betriebssystem zu Betriebssystem unterscheiden. Eine Verschlechterung des Bildes aufgrund von Skalierung können Sie vermeiden, indem Sie Bilder in mehreren Größen zum `bitmaps`-Array hinzufügen. Wenn Sie mehrere Bilder hinzugefügt haben, wird von AIR automatisch die Größe ausgewählt, die der gegenwärtigen Anzeigegröße des Taskleisten-Symbols am nächsten kommt. Eine Skalierung wird nur bei Bedarf vorgenommen. Im folgenden Beispiel wird das Bild für ein Taskleisten-Symbol mit zwei Bildern festgelegt:

```
NativeApplication.nativeApplication.icon.bitmaps =
            [bmp16x16.bitmapData, bmp128x128.bitmapData];
```

Um das Bild des Symbols zu ändern, weisen Sie der `bitmaps`-Eigenschaft ein Array zu, das das neue Bild oder die neuen Bilder enthält. Sie können das Symbol animieren, indem Sie das Bild auf ein `enterFrame`- oder `timer`-Ereignis hin ändern.

Weisen Sie der `bitmaps`-Eigenschaft ein leeres Array zu, um das Symbol unter Windows oder Linux aus dem Infobereich zu entfernen oder unter Mac OS X das Standarderscheinungsbild des Symbols wiederherzustellen:

```
NativeApplication.nativeApplication.icon.bitmaps = [];
```

Dock-Symbole

Adobe AIR 1.0 und höher

AIR unterstützt Dock-Symbole, wenn die `NativeApplication.supportsDockIcon`-Eigenschaft den Wert `true` hat. Die `NativeApplication.nativeApplication.icon`-Eigenschaft stellt das Anwendungssymbol im Dock dar (kein Fenster-Dock-Symbol).

Hinweis: *Unter Mac OS X wird das Ändern von Fenstersymbolen im Dock von AIR nicht unterstützt. Außerdem wirken sich die Änderungen an einem Anwendungssymbol im Dock nur aus, während die Anwendung ausgeführt wird; das normale Erscheinungsbild des Symbols wird wiederhergestellt, wenn die Anwendung beendet wird.*

Dock-Symbolmenüs

Adobe AIR 1.0 und höher

Sie können dem standardmäßigen Dock-Menü Befehle hinzufügen, indem Sie ein NativeMenu-Objekt, das die Befehle enthält, erstellen und der `NativeApplication.nativeApplication.icon.menu`-Eigenschaft zuweisen. Elemente im Menü werden über den standardmäßigen Dock-Symbolmenüelementen angezeigt.

Springen des Dock-Symbols

Adobe AIR 1.0 und höher

Durch Aufrufen der `NativeApplication.nativeApplication.icon.bounce()`-Methode können Sie das Dock-Symbol zum Springen veranlassen. Wenn Sie den `bounce() priority`-Parameter auf „informational“ einstellen, springt das Symbol einmal. Wenn Sie es auf „critical“ einstellen, springt das Symbol, bis der Benutzer die Anwendung aktiviert. Konstanten für den `priority`-Parameter sind in der NotificationType-Klasse definiert:

Hinweis: *Ist die Anwendung bereits aktiv, springt das Symbol nicht.*

Dock-Symbolereignisse

Adobe AIR 1.0 und höher

Wird auf das Dock-Symbol geklickt, löst das NativeApplication-Objekt ein `invoke`-Ereignis aus. Falls die Anwendung nicht ausgeführt wird, wird sie vom System gestartet. Andernfalls wird das `invoke`-Ereignis an die ausgeführte Anwendungsinstanz gesendet.

Infobereich-Symbole

Adobe AIR 1.0 und höher

AIR unterstützt Infobereich-Symbole, wenn `NativeApplication.supportsSystemTrayIcon` auf `true` eingestellt ist; gegenwärtig ist dies nur unter Windows und den meisten Linux-Distributionen der Fall. Unter Windows und Linux werden Infobereich-Symbole im Infobereich der Taskleiste angezeigt. Standardmäßig wird kein Symbol angezeigt. Um ein Symbol anzuzeigen, weisen Sie der `bitmaps`-Eigenschaft ein Array zu, das BitmapData-Objekte enthält: Um das Bild des Symbols zu ändern, weisen Sie der `bitmaps`-Eigenschaft ein Array zu, das die neuen Bilder enthält. Stellen Sie `bitmaps` auf `null` ein, um das Symbol zu entfernen.

Infobereich-Symbolmenüs

Adobe AIR 1.0 und höher

Sie können dem Infobereich-Symbol ein Menü hinzufügen, indem Sie ein NativeMenu-Objekt erstellen und es der `NativeApplication.nativeApplication.icon.menu`-Eigenschaft zuweisen (vom Betriebssystem wird kein Standardmenü bereitgestellt). Klicken Sie mit der rechten Maustaste auf das Symbol, um das Infobereich-Symbolmenü aufzurufen.

QuickInfos zu Infobereich-Symbolen

Adobe AIR 1.0 und höher

Fügen Sie einem Symbol eine QuickInfo hinzu, indem Sie die tooltip-Eigenschaft wie folgt einstellen:

```
NativeApplication.nativeApplication.icon.tooltip = "Application name";
```

Ereignisse von Infobereich-Symbolen

Adobe AIR 1.0 und höher

Das SystemTrayIcon-Objekt, auf das die NativeApplication.nativeApplication.icon-Eigenschaft verweist, löst ein „ScreenMouseEvent“ für `click`-, `mouseDown`-, `mouseUp`-, `rightClick`-, `rightMouseDown`- und `rightMouseUp`-Ereignisse aus. Mithilfe dieser Ereignisse und einem Symbolmenü können Sie Benutzern die Interaktion mit Ihrer Anwendung ermöglichen, wenn sie keine sichtbaren Fenster hat.

Beispiel: Erstellen einer Anwendung ohne Fenster

Adobe AIR 1.0 und höher

Im folgenden Beispiel wird eine AIR-Anwendung erstellt, die über ein Infobereich-Symbol, aber keine sichtbaren Fenster verfügt. (Die `visible`-Eigenschaft der Anwendung darf im Anwendungsdeskriptor nicht auf `true` gesetzt werden, andernfalls ist das Fenster beim Starten der Anwendung sichtbar.)

```
package
{
    import flash.display.Loader;
    import flash.display.NativeMenu;
    import flash.display.NativeMenuItem;
    import flash.display.NativeWindow;
    import flash.display.Sprite;
    import flash.desktop.DockIcon;
    import flash.desktop.SystemTrayIcon;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.desktop.NativeApplication;

    public class SysTrayApp extends Sprite
    {
        public function SysTrayApp():void{
            NativeApplication.nativeApplication.autoExit = false;
            var icon:Loader = new Loader();
            var iconMenu:NativeMenu = new NativeMenu();
            var exitCommand:NativeMenuItem = iconMenu.addItem(new NativeMenuItem("Exit"));
```

```
                exitCommand.addEventListener(Event.SELECT, function(event:Event):void {
                    NativeApplication.nativeApplication.icon.bitmaps = [];
                    NativeApplication.nativeApplication.exit();
                });

            if (NativeApplication.supportsSystemTrayIcon) {
                NativeApplication.nativeApplication.autoExit = false;
                icon.contentLoaderInfo.addEventListener(Event.COMPLETE, iconLoadComplete);
                icon.load(new URLRequest("icons/AIRApp_16.png"));

                var systray:SystemTrayIcon =
                    NativeApplication.nativeApplication.icon as SystemTrayIcon;
                systray.tooltip = "AIR application";
                systray.menu = iconMenu;
            }

            if (NativeApplication.supportsDockIcon){
                icon.contentLoaderInfo.addEventListener(Event.COMPLETE,iconLoadComplete);
                icon.load(new URLRequest("icons/AIRApp_128.png"));
                var dock:DockIcon = NativeApplication.nativeApplication.icon as DockIcon;
                dock.menu = iconMenu;
            }
        }

        private function iconLoadComplete(event:Event):void
        {
            NativeApplication.nativeApplication.icon.bitmaps =
                [event.target.content.bitmapData];
        }
    }
}
```

Hinweis: *Wenn Sie mit der Flex WindowedApplication-Komponente arbeiten, müssen Sie das* `visible`*-Attribut des WindowedApplication-Tags auf* `false` *setzen. Dieses ersetzt die Einstellung im Anwendungsdeskriptor.*

Hinweis: *Bei diesem Beispiel wird davon ausgegangen, dass es die Bilddateien* `AIRApp_16.png` *und* `AIRApp_128.png` *in einem* `icons`*-Unterverzeichnis der Anwendung gibt. (Beispieldateien für Symbole, die Sie in Ihren Projektordner kopieren können, sind im AIR SDK enthalten.)*

Taskleisten-Symbole und -Schaltflächen für Fenster

Adobe AIR 1.0 und höher

Symboldarstellungen von Fenstern werden in der Regel im Fensterbereich einer Taskleiste oder eines Docks angezeigt, um Benutzern einen einfachen Zugriff auf Fenster im Hintergrund oder minimierte Fenster zu ermöglichen. Im Mac OS X-Dock wird sowohl ein Symbol für die Anwendung als auch ein Symbol für jedes minimierte Fenster angezeigt. In der Taskleiste von Microsoft Windows und Linux wird eine Schaltfläche angezeigt, die das Programmsymbol und den Programmnamen für jedes Fenster vom Typ „normal“ in der Anwendung enthält.

Hervorheben der Fensterschaltfläche in der Taskleiste

Adobe AIR 1.0 und höher

Wenn sich ein Fenster im Hintergrund befindet, können Sie den Benutzer informieren, dass im Zusammenhang mit diesem Fenster ein Ereignis aufgetreten ist. Unter Mac OS X können Sie den Benutzer durch ein Springen des Anwendungssymbols im Dock informieren (siehe „Springen des Dock-Symbols“ auf Seite 690). Unter Windows und Linux können Sie die Fensterschaltfläche in der Taskleiste hervorheben, indem Sie die `notifyUser()`-Methode der NativeWindow-Instanz aufrufen. Mit dem an die Methode übergebenen `type`-Parameter wird die Dringlichkeit der Benachrichtigung bestimmt:

- `NotificationType.CRITICAL`: Das Fenster-Symbol blinkt, bis der Benutzer das Fenster in den Vordergrund stellt.
- `NotificationType.INFORMATIONAL`: Das Fenster-Symbol wird durch Änderung der Farbe hervorgehoben.

  Hinweis: *Unter Linux werden nur informative Benachrichtigungen unterstützt. Wenn Sie einen Wert des einen oder anderen Typs an die `notifyUser()`-Funktion übergeben, wird derselbe Effekt erzielt.*

  Mit der folgenden Anweisung wird die Schaltfläche eines Fensters in der Taskleiste hervorgehoben:

```
stage.nativeWindow.notifyUser(NotificationType.CRITICAL);
```

  Wird die `NativeWindow.notifyUser()`-Methode unter einem Betriebssystem aufgerufen, dass die Benachrichtigung auf Fensterebene nicht unterstützt, hat dieser Aufruf keine Wirkung. Mithilfe der `NativeWindow.supportsNotification`-Eigenschaft können Sie feststellen, ob diese Art der Benachrichtigung unterstützt wird.

Erstellen von Fenstern ohne Schaltflächen oder Symbolen in der Taskleiste

Adobe AIR 1.0 und höher

Unter Windows werden Fenster, die mit den Typen *utility* oder *lightweight* erstellt wurden, nicht in der Taskleiste angezeigt. Unsichtbare Fenster werden ebenfalls nicht in der Taskleiste angezeigt.

Da das erste Fenster zwangsläufig den Typ *normal* aufweist, müssen Sie dieses erste Fenster schließen oder unsichtbar belassen, wenn Sie eine Anwendung erstellen möchten, deren Fenster nicht in der Taskleiste angezeigt werden. Um alle Fenster in der Anwendung zu schließen, ohne die Anwendung zu beenden, müssen Sie die `autoExit`-Eigenschaft des NativeApplication-Objekts auf `false` einstellen, bevor Sie das letzte Fenster schließen. Um zu verhindern, dass das erste Fenster jemals sichtbar wird, fügen Sie `<visible>false</visible>` zum `<initalWindow>`-Element der Anwendungsdeskriptordatei hinzu. Außerdem sollten Sie die `visible`-Eigenschaft nicht auf `true` einstellen oder die `activate()`-Methode des Fensters aufrufen.

Stellen Sie in neuen, von der Anwendung geöffneten Fenstern die `type`-Eigenschaft des NativeWindowInitOption-Objekts, das an den Fensterkonstruktor übergeben wird, auf `NativeWindowType.UTILITY` oder `NativeWindowType.LIGHTWEIGHT` ein.

Unter Mac OS X werden minimierte Fenster in der Dock-Taskleiste angezeigt. Sie können die Anzeige des minimierten Symbols vermeiden, indem Sie das Fenster nicht minimieren, sondern ausblenden. Im folgenden Beispiel wird auf ein `nativeWindowDisplayState`-Änderungsereignis gewartet; es wird abgebrochen, falls das Fenster minimiert wird. Stattdessen stellt die Prozedur die `visible`-Eigenschaft des Fensters auf `false` ein:

```
 private function preventMinimize(event:NativeWindowDisplayStateEvent):void{
    if(event.afterDisplayState == NativeWindowDisplayState.MINIMIZED){
        event.preventDefault();
        event.target.visible = false;
    }
}
```

Wird ein Fenster im Mac OS X-Dock minimiert angezeigt, wenn Sie die `visible`-Eigenschaft auf `false` einstellen, dann wird das Dock-Symbol nicht entfernt. Ein Benutzer kann nach wie vor auf das Symbol klicken, um das Fenster wieder anzuzeigen.

Kapitel 38: Arbeiten mit dem Dateisystem

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash® Player bietet grundlegende Möglichkeiten zum Lesen und Schreiben von Dateien über die FileReference-Klasse. Aus Sicherheitsgründen muss der Benutzer bei der Ausführung in Flash Player die entsprechende Berechtigung erteilen, bevor Sie eine Datei lesen oder schreiben können.

Adobe® AIR® bietet umfassenderen Zugriff auf das Dateisystem des Hostcomputers als Flash Player. Mit der AIR-Dateisystem-API können Sie Verzeichnisse und Dateien aufrufen, erstellen und verwalten, in Dateien schreiben usw.

Verwandte Hilfethemen

flash.net.FileReference

flash.net.FileReferenceList

flash.filesystem.File

flash.filesystem.FileStream

Verwenden der FileReference-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein FileReference-Objekt repräsentiert eine Datendatei auf einem Client oder Server. Mit den Methoden der FileReference-Klasse kann Ihre Anwendung Datendateien lokal laden und speichern sowie Dateidaten zwischen Remote-Servern übertragen.

Die FileReference-Klasse bietet zwei verschiedene Verfahren zum Laden, Übertragen und Speichern von Datendateien. Seit ihrer Einführung enthält die FileReference-Klasse die Methoden `browse()`, `upload()` und `download()`. Verwenden Sie die `browse()`-Methode, um dem Benutzer die Möglichkeit zu geben, eine Datei auszuwählen. Verwenden Sie die `upload()`-Methode, um die Daten der Datei auf einen Remoteserver zu übertragen. Verwenden Sie die `download()`-Methode, um diese Daten vom Server abzurufen und in einer lokalen Datei zu speichern. Ab Flash Player 10 und Adobe AIR 1.5 enthält die FileReference-Klasse die Methoden `load()` und `save()`. Mit den Methoden `load()` und `save()` können Sie lokale Dateien auch direkt aufrufen und speichern. Die Verwendung dieser Methoden ähnelt der Verwendung der gleichnamigen Methoden der URLLoader- und Loader-Klassen.

Hinweis: *Die File-Klasse, die die FileReference-Klasse erweitert, und die FileStream-Klasse bieten zusätzliche Funktionen für die Arbeit mit Dateien und dem lokalen Dateisystem. Die File-Klasse und die FileStream-Klasse werden nur in AIR, nicht in Flash Player unterstützt.*

FileReference-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jedes FileReference-Objekt stellt eine einzelne Datendatei auf dem lokalen Computer dar. Die Eigenschaften der FileReference-Klasse umfassen Informationen zu Dateigröße, Dateityp, Dateiname, Dateinamenerweiterung, Ersteller, Erstellungs- sowie Änderungsdatum.

Hinweis: *Die `creator`-Eigenschaft wird nur unter MacOS unterstützt. Auf allen anderen Plattformen wird `null` zurückgegeben.*

Hinweis: *Die `extension`-Eigenschaft wird nur in Adobe AIR unterstützt.*

Eine Instanz der FileReference-Klasse kann mit zwei Verfahren erstellt werden:

- Verwenden Sie den `new`-Operator, wie im folgenden Codebeispiel gezeigt:

```
import flash.net.FileReference;
var fileRef:FileReference = new FileReference();
```

- Rufen Sie die Methode `FileReferenceList.browse()` auf. Dadurch wird ein Dialogfeld geöffnet, in dem der Benutzer aufgefordert wird, eine oder mehrere Dateien für den Upload auszuwählen. Nachdem der Benutzer eine oder mehrere Dateien ausgewählt hat, wird ein Array aus FileReference-Objekten erstellt.

Nach der Erstellung eines FileReference-Objekts haben Sie folgende Möglichkeiten:

- Rufen Sie die `FileReference.browse()`-Methode auf. Dadurch wird ein Dialogfeld geöffnet, in dem der Benutzer aufgefordert wird, genau eine Datei im lokalen Dateisystem auszuwählen. Dies erfolgt normalerweise vor einem späteren Aufruf der Methode `FileReference.upload()` oder `FileReference.load()`. Rufen Sie die `FileReference.upload()`-Methode auf, um die Datei auf einen Remoteserver hochzuladen. Rufen Sie die `FileReference.load()`-Methode auf, um eine lokale Datei zu öffnen.
- Rufen Sie die `FileReference.download()`-Methode auf. Die `download`-Methode öffnet ein Dialogfeld, in dem der Benutzer ein Verzeichnis zum Speichern einer neuen Datei auswählen kann. Dann werden Daten vom Server heruntergeladen und in der neuen Datei gespeichert.
- Rufen Sie die `FileReference.load()`- Methode auf. Diese Methode beginnt, Daten aus einer vor Verwendung der `browse()`-Methode ausgewählten Datei zu laden. Die `load()`-Methode kann erst aufgerufen werden, wenn der `browse()`-Vorgang abgeschlossen ist (der Benutzer eine Datei auswählt).
- Rufen Sie die `FileReference.save()`-Methode auf. Diese Methode öffnet ein Dialogfeld und fordert den Benutzer auf, einen einzelnen Dateispeicherort im lokalen Dateisystem auszuwählen. Die Daten werden dann am angegebenen Speicherort gespeichert.

Hinweis: *Es kann immer nur jeweils eine `browse()`-, `download()`- oder `save`-Aktion durchgeführt werden, da immer nur ein Dialogfeld geöffnet sein kann.*

Die Eigenschaften des FileReference-Objekts, wie `name`, `size` oder `modificationDate`, werden erst nach einem der folgenden Vorgänge definiert:

- Die `FileReference.browse()`-Methode oder die `FileReferenceList.browse()`-Methode wurde aufgerufen und der Benutzer hat eine Datei im Dialogfeld ausgewählt.
- Die `FileReference.download()`-Methode wurde aufgerufen und der Benutzer hat ein neues Verzeichnis für die Datei im Dialogfeld angegeben.

Hinweis: *Während eines Downloads ist nur die Eigenschaft `FileReference.name` mit Daten gefüllt, bevor der Download abgeschlossen ist. Nachdem die Datei heruntergeladen wurde, stehen alle Eigenschaften zur Verfügung.*

Während Aufrufe der Methode `FileReference.browse()`, `FileReferenceList.browse()`, `FileReference.download()`, `FileReference.load()` oder `FileReference.save()` ausgeführt werden, setzen die meisten Player die Wiedergabe der SWF-Datei fort, einschließlich Ereignisauslösung und Codeausführung.

Bei Upload- und Download-Vorgängen kann mit einer SWF-Datei nur auf Dateien in der eigenen Domäne und in den in einer Richtliniendatei angegebenen Domänen zugegriffen werden. Sie müssen auf dem Server mit der Datei eine Richtliniendatei ablegen, falls sich dieser Server nicht in derselben Domäne wie die SWF-Datei befindet, die den Upload oder Download eingeleitet hat.

Siehe FileReference.

Laden von Daten aus Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `FileReference.load()`-Methode können Sie Daten aus einer lokalen Datei in den Arbeitsspeicher laden.

Hinweis: *Der Code muss zuerst die `FileReference.browse()`-Methode aufrufen, damit der Benutzer die zu ladende Datei auswählen kann. Diese Einschränkung gilt nicht für Inhalt, der in der Anwendungs-Sicherheits-Sandbox in Adobe AIR ausgeführt wird.*

Die Rückgabe von der `FileReference.load()`-Methode erfolgt direkt nach dem Aufrufen, die Daten, die geladen werden, stehen jedoch nicht sofort zur Verfügung. Das FileReference-Objekt löst Ereignisse aus, um bei jedem Schritt des Ladevorgangs Listener-Methoden aufzurufen.

Das FileReference-Objekt löst beim Speichern der Datei die folgenden Ereignisse aus.

- `open`-Ereignis (`Event.OPEN`): Wird ausgelöst, wenn der Ladevorgang beginnt.
- `progress`-Ereignis (`ProgressEvent.PROGRESS`): Wird regelmäßig ausgelöst, während Daten-Bytemengen aus der Datei gelesen werden.
- `complete`-Ereignis (`Event.COMPLETE`): Wird ausgelöst, wenn das Laden erfolgreich abgeschlossen wurde.
- `ioError`-Ereignis (`IOErrorEvent.IO_ERROR`): Wird ausgelöst, wenn der Ladevorgang fehlschlägt, weil beim Öffnen der Datei oder beim Lesen von Daten aus der Datei ein Eingabe-/Ausgabefehler auftritt.

Nachdem das FileReference-Objekt das complete-Ereignis ausgelöst hat, kann auf die geladenen Daten als ein ByteArray in der `data`-Eigenschaft des FileReference-Objekts zugegriffen werden.

Im folgenden Codebeispiel wird gezeigt, wie der Benutzer zur Auswahl einer Datei aufgefordert wird und wie die Daten dann aus dieser Datei in den Arbeitsspeicher geladen werden:

```
package
{
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.FileFilter;
    import flash.net.FileReference;
    import flash.net.URLRequest;
    import flash.utils.ByteArray;

    public class FileReferenceExample1 extends Sprite
    {
        private var fileRef:FileReference;
        public function FileReferenceExample1()
        {
            fileRef = new FileReference();
            fileRef.addEventListener(Event.SELECT, onFileSelected);
            fileRef.addEventListener(Event.CANCEL, onCancel);
            fileRef.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
            fileRef.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
                        onSecurityError);
            var textTypeFilter:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)",
                        "*.txt;*.rtf");
            fileRef.browse([textTypeFilter]);
        }
        public function onFileSelected(evt:Event):void
        {
            fileRef.addEventListener(ProgressEvent.PROGRESS, onProgress);
            fileRef.addEventListener(Event.COMPLETE, onComplete);
            fileRef.load();
        }

        public function onProgress(evt:ProgressEvent):void
        {
            trace("Loaded " + evt.bytesLoaded + " of " + evt.bytesTotal + " bytes.");
        }

        public function onComplete(evt:Event):void
        {
            trace("File was successfully loaded.");
            trace(fileRef.data);
        }

        public function onCancel(evt:Event):void
        {
            trace("The browse request was canceled by the user.");
        }

        public function onIOError(evt:IOErrorEvent):void
        {
            trace("There was an IO Error.");
        }
        public function onSecurityError(evt:Event):void
        {
            trace("There was a security error.");
        }
    }
}
```

Der Beispielcode erstellt zuerst das FileReference-Objekt mit den Namen `fileRef` und ruft dann dessen `browse()`-Methode auf. Die `browse`-Methode öffnet ein Dialogfeld, das den Benutzer zum Auswählen einer Datei auffordert. Wenn eine Datei ausgewählt wird, ruft der Code die `onFileSelected()`-Methode auf. Diese Methode fügt Listener für die `progress`- und `complete`-Ereignisse hinzu und ruft dann die `load()`-Methode des FileReference-Objekts auf. Die anderen Prozedurmethoden im Beispiel geben Meldungen aus, um den Fortschritt des Ladevorgangs zu melden. Wenn der Ladevorgang abgeschlossen ist, zeigt die Anwendung den Inhalt der geladenen Datei mithilfe der `trace()`-Methode an.

In Adobe AIR bietet die FileStream-Klasse zusätzliche Funktionen zum Lesen von Daten aus einer lokalen Datei. Lesen Sie dazu „Lese- und Schreibvorgänge in Dateien“ auf Seite 735.

Speichern von Daten in lokalen Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `FileReference.save()`-Methode können Sie Daten in einer lokalen Datei speichern. Zunächst wird ein Dialogfeld geöffnet, in dem der Benutzer einen neuen Dateinamen sowie das Verzeichnis zum Speichern der Datei eingeben kann. Nachdem der Benutzer den Dateinamen und den Speicherort ausgewählt hat, werden die Daten in die neue Datei geschrieben. Wenn die Datei erfolgreich gespeichert wurde, werden die Eigenschaften des FileReference-Objekts mit den Eigenschaften der lokalen Datei gefüllt.

Hinweis: *Im Code kann die `FileReference.save()`-Methode nur als Reaktion auf ein vom Benutzer initiiertes Ereignis aufgerufen werden, wie beispielsweise ein Mausklick- oder Tastendruck-Ereignis. Andernfalls wird ein Fehler ausgegeben. Diese Einschränkung gilt nicht für Inhalt, der in der Anwendungs-Sicherheits-Sandbox in Adobe AIR ausgeführt wird.*

Die Rückgabe der `FileReference.save()`-Methode erfolgt unmittelbar nach ihrem Aufruf. Dann löst das FileReference-Objekt Ereignisse aus, um bei jedem Schritt des Speichervorgangs Listener-Methoden aufzurufen.

Das FileReference-Objekt löst beim Speichern der Datei die folgenden Ereignisse aus:

- `select`-Ereignis (`Event.SELECT`): Wird ausgelöst, wenn der Benutzer das Verzeichnis und den Dateinamen für die neue Datei festlegt, die gespeichert werden soll.
- `cancel`-Ereignis (`Event.CANCEL`): Wird ausgelöst, wenn der Benutzer im Dialogfeld auf die Schaltfläche „Abbrechen“ klickt.
- `open`-Ereignis (`Event.OPEN`): Wird ausgelöst, wenn der Speichervorgang beginnt.
- `progress`-Ereignis (`ProgressEvent.PROGRESS`): Wird regelmäßig ausgelöst, während Daten-Bytemengen in der Datei gespeichert werden.
- `complete`-Ereignis (`Event.COMPLETE`): Wird ausgelöst, wenn das Speichern erfolgreich abgeschlossen wurde.
- `ioError`-Ereignis (`IOErrorEvent.IO_ERROR`): Wird ausgelöst, wenn der Speichervorgang fehlschlägt, weil beim Versuch, die Daten in der Datei zu speichern, ein Eingabe-/Ausgabefehler auftritt.

Der im `data`-Parameter der `FileReference.save()`-Methode übergebene Objekttyp bestimmt, wie die Daten in die Datei geschrieben werden:

- Bei einem String-Wert werden die Daten als Textdatei in UTF-8-Kodierung gespeichert.
- Bei einem XML-Objekt werden die Daten in eine Datei im XML-Format geschrieben, wobei die gesamte Formatierung beibehalten wird.
- Bei einem ByteArray-Objekt wird der Inhalt ohne Konvertierung direkt in die Datei geschrieben.

- Bei allen anderen Objekten ruft die `FileReference.save()`-Methode die `toString()`-Methode des Objekts auf und speichert dann den resultierenden String-Wert in einer Textdatei in UTF-8-Kodierung. Wenn die `toString()`-Methode des Objekts nicht aufgerufen werden kann, wird ein Fehler ausgegeben.

Wenn der `data`-Parameter den Wert `null` hat, wird ein Fehler ausgegeben.

Der folgende Code erweitert das vorstehende Beispiel für die `FileReference.load()`-Methode. Nachdem die Daten aus der Datei gelesen wurden, wird der Benutzer in diesem Beispiel zur Angabe eines Dateinamens aufgefordert. Dann werden die Daten in einer neuen Datei gespeichert:

```
package
{
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.FileFilter;
    import flash.net.FileReference;
    import flash.net.URLRequest;
    import flash.utils.ByteArray;

    public class FileReferenceExample2 extends Sprite
    {
        private var fileRef:FileReference;
        public function FileReferenceExample2()
        {
            fileRef = new FileReference();
            fileRef.addEventListener(Event.SELECT, onFileSelected);
            fileRef.addEventListener(Event.CANCEL, onCancel);
            fileRef.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
            fileRef.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
                        onSecurityError);
            var textTypeFilter:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)",
                        "*.txt;*.rtf");
            fileRef.browse([textTypeFilter]);
        }
        public function onFileSelected(evt:Event):void
        {
            fileRef.addEventListener(ProgressEvent.PROGRESS, onProgress);
            fileRef.addEventListener(Event.COMPLETE, onComplete);
            fileRef.load();
        }

        public function onProgress(evt:ProgressEvent):void
        {
            trace("Loaded " + evt.bytesLoaded + " of " + evt.bytesTotal + " bytes.");
        }
        public function onCancel(evt:Event):void
        {
            trace("The browse request was canceled by the user.");
        }
        public function onComplete(evt:Event):void
        {
            trace("File was successfully loaded.");
            fileRef.removeEventListener(Event.SELECT, onFileSelected);
            fileRef.removeEventListener(ProgressEvent.PROGRESS, onProgress);
            fileRef.removeEventListener(Event.COMPLETE, onComplete);
            fileRef.removeEventListener(Event.CANCEL, onCancel);
            saveFile();
```

```
        }
        public function saveFile():void
        {
            fileRef.addEventListener(Event.SELECT, onSaveFileSelected);
            fileRef.save(fileRef.data,"NewFileName.txt");
        }

        public function onSaveFileSelected(evt:Event):void
        {
            fileRef.addEventListener(ProgressEvent.PROGRESS, onSaveProgress);
            fileRef.addEventListener(Event.COMPLETE, onSaveComplete);
            fileRef.addEventListener(Event.CANCEL, onSaveCancel);
        }

        public function onSaveProgress(evt:ProgressEvent):void
        {
            trace("Saved " + evt.bytesLoaded + " of " + evt.bytesTotal + " bytes.");
        }

        public function onSaveComplete(evt:Event):void
        {
            trace("File saved.");
            fileRef.removeEventListener(Event.SELECT, onSaveFileSelected);
            fileRef.removeEventListener(ProgressEvent.PROGRESS, onSaveProgress);
            fileRef.removeEventListener(Event.COMPLETE, onSaveComplete);
            fileRef.removeEventListener(Event.CANCEL, onSaveCancel);
        }

        public function onSaveCancel(evt:Event):void
        {
            trace("The save request was canceled by the user.");
        }

        public function onIOError(evt:IOErrorEvent):void
        {
            trace("There was an IO Error.");
        }
        public function onSecurityError(evt:Event):void
        {
            trace("There was a security error.");
        }
    }
}
```

Wenn alle Daten aus der Datei geladen wurden, ruft der Code die `onComplete()`-Methode auf. Die `onComplete()`-Methode entfernt die Listener für die Ladeereignisse und ruft dann die `saveFile()`-Methode auf. Die `saveFile()`-Methode ruft die `FileReference.save()`-Methode auf. Die `FileReference.save()`-Methode öffnet ein neues Dialogfeld, in dem der Benutzer einen neuen Dateinamen eingeben und den Speicherort der Datei auswählen kann. Die restlichen Ereignis-Listener-Methoden verfolgen den Fortschritt des Speichervorgangs bis zum Abschluss.

In Adobe AIR bietet die FileStream-Klasse zusätzliche Funktionen zum Schreiben von Daten in eine lokale Datei. Lesen Sie dazu „Lese- und Schreibvorgänge in Dateien“ auf Seite 735.

Hochladen von Dateien auf einen Server

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Hochladen von Dateien auf einen Server rufen Sie zunächst die `browse()`-Methode auf, damit ein Benutzer eine oder mehrere Dateien auswählen kann. Dann, nachdem die `FileReference.upload()`-Methode aufgerufen wurde, wird die ausgewählte Datei an den Server übertragen. Wählt der Benutzer mehrere Dateien mit der Methode `FileReferenceList.browse()` aus, erstellt Flash Player ein Array der ausgewählten Dateien mit der Bezeichnung `FileReferenceList.fileList`. Sie können dann die `FileReference.upload()`-Methode verwenden, um die Dateien einzeln hochzuladen.

Hinweis: *Mit der `FileReference.browse()`-Methode können Sie nur einzelne Dateien hochladen. Um einem Benutzer das Hochladen mehrerer Dateien zu ermöglichen, verwenden Sie die `FileReferenceList.browse()`-Methode.*

Standardmäßig können Benutzer auf dem lokalen Computer einen beliebigen Dateityp im Dialogfeld zur Dateiauswahl auswählen. Entwickler können mit der FileFilter-Klasse und durch Übergeben eines Arrays mit Dateifilterinstanzen an die `browse()`-Methode einen oder mehrere benutzerdefinierte Filter für Dateitypen angeben:

```
var imageTypes:FileFilter = new FileFilter("Images (*.jpg, *.jpeg, *.gif, *.png)", "*.jpg;
*.jpeg; *.gif; *.png");
var textTypes:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)", "*.txt; *.rtf");
var allTypes:Array = new Array(imageTypes, textTypes);
var fileRef:FileReference = new FileReference();
fileRef.browse(allTypes);
```

Nachdem der Benutzer die Dateien ausgewählt und im Dialogfeld zur Dateiauswahl auf die Schaltfläche zum Öffnen der Datei geklickt hat, wird das `Event.SELECT`-Ereignis ausgelöst. Wird die `FileReference.browse()`-Methode zur Auswahl einer hochzuladenden Datei verwendet, sendet der folgende Code die Datei an einen Webserver:

```
var fileRef:FileReference = new FileReference();
fileRef.addEventListener(Event.SELECT, selectHandler);
fileRef.addEventListener(Event.COMPLETE, completeHandler);
try
{
    var success:Boolean = fileRef.browse();
}
catch (error:Error)
{
    trace("Unable to browse for files.");
}
function selectHandler(event:Event):void
{
    var request:URLRequest = new URLRequest("http://www.[yourdomain].com/fileUploadScript.cfm")
    try
    {
        fileRef.upload(request);
    }
    catch (error:Error)
    {
        trace("Unable to upload file.");
    }
}
function completeHandler(event:Event):void
{
    trace("uploaded");
}
```

Mit der `FileReference.upload()`-Methode können Sie Daten an den Server senden, indem Sie die Eigenschaften `URLRequest.method` und `URLRequest.data` verwenden, um Variablen mit den Methoden `POST` oder `GET` zu senden.

Wenn Sie versuchen, eine Datei mit der `FileReference.upload()`-Methode hochzuladen, werden die folgenden Ereignisse ausgelöst:

- `open`-Ereignis (`Event.OPEN`): Wird ausgelöst, wenn der Uploadvorgang beginnt.
- `progress`-Ereignis (`ProgressEvent.PROGRESS`): Wird regelmäßig ausgelöst, während Daten-Bytemengen aus der Datei hochgeladen werden.
- `complete`-Ereignis (`Event.COMPLETE`): Wird ausgelöst, wenn der Uploadvorgang erfolgreich abgeschlossen wurde.
- `httpStatus`-Ereignis (`HTTPStatusEvent.HTTP_STATUS`): Wird ausgelöst, wenn ein Upload-Vorgang aufgrund eines HTTP-Fehlers fehlschlägt.
- `httpResponseStatus`-Ereignis (`HTTPStatusEvent.HTTP_RESPONSE_STATUS`): Wird ausgelöst, wenn ein Aufruf der `upload()`- oder `uploadUnencoded()`-Methode versucht, über HTTP auf Daten zuzugreifen, und Adobe AIR den Statuscode für die Anforderung erkennen und zurückgeben kann.
- `securityError`-Ereignis (`SecurityErrorEvent.SECURITY_ERROR`): Wird ausgelöst, wenn ein Upload-Vorgang aufgrund einer Sicherheitsverletzung fehlschlägt.
- `uploadCompleteData`-Ereignis `DataEvent.UPLOAD_COMPLETE_DATA`): Wird ausgelöst, nachdem Daten vom Server nach einem erfolgreichen Upload empfangen wurden.
- `ioError`-Ereignis (`IOErrorEvent.IO_ERROR`): Wird ausgelöst, wenn der Upload-Vorgang aus einem der folgenden Gründe fehlschlägt:
  - Während des Lese-, Schreib- oder Übertragungsvorgangs durch Flash Player ist ein Eingabe/Ausgabe-Fehler aufgetreten.
  - Die SWF-Datei versuchte, eine Datei auf einen Server hochzuladen, der eine Authentifizierung erfordert (z. B. anhand eines Benutzernamens und eines Kennworts). Während des Uploads stellt Flash Player keine Möglichkeiten für die Eingabe von Kennwörtern durch die Benutzer bereit.
  - Der Parameter `url` enthält ein ungültiges Protokoll. Die `FileReference.upload()`-Methode muss entweder HTTP oder HTTPS verwenden.

Flash Player bietet keine Unterstützung für Server, die eine Authentifizierung erfordern. Nur bei SWF-Dateien, die in einem Browser mithilfe des Browser-Plug-Ins oder des Microsoft ActiveX®-Steuerelements ausgeführt werden, kann der Benutzer in einem Dialogfeld zur Eingabe eines Benutzernamens und eines Kennworts zur Authentifizierung aufgefordert werden. Dies gilt jedoch nur für Downloadvorgänge. Upload-Vorgänge mit Plug-Ins oder ActiveX-Steuerelementen bzw. Upload/Download-Vorgänge mit dem eigenständigen oder externen Player schlagen fehl.

Wenn Sie ein Serverskript in ColdFusion erstellen möchten, um einen Datei-Upload von Flash Player zu akzeptieren, können Sie Code wie den Folgenden verwenden:

```
<cffile action="upload" filefield="Filedata" destination="#ExpandPath('./')#"
nameconflict="OVERWRITE" />
```

Dieser ColdFusion-Code lädt die von Flash Player gesendete Datei hoch und speichert sie im gleichen Verzeichnis wie die ColdFusion-Vorlage. Dabei wird eine Datei mit dem gleichen Namen überschrieben. Das vorangegangene Codebeispiel zeigt den mindestens erforderlichen Code, der zum Akzeptieren eines Datei-Uploads erforderlich ist. Dieses Skript sollte in einer Produktionsumgebung nicht verwendet werden. Im Idealfall fügen Sie eine Datenvalidierung hinzu, um sicherzustellen, dass Benutzer nur akzeptierte Dateitypen hochladen, beispielsweise eine Bilddatei anstelle eines potenziell gefährlichen serverseitigen Skripts.

Der folgende Code demonstriert Datei-Uploads mithilfe von PHP und umfasst eine Datenvalidierung. Das Skript beschränkt die Anzahl der hochgeladenen Dateien im Upload-Verzeichnis auf 10, und stellt sicher, dass die Datei kleiner als 200 KB ist. Weiterhin werden das Hochladen und die Speicherung im Dateisystem auf Dateien des Typs JPEG, GIF oder PNG beschränkt.

```
<?php
$MAXIMUM_FILESIZE = 1024 * 200; // 200KB
$MAXIMUM_FILE_COUNT = 10; // keep maximum 10 files on server
echo exif_imagetype($_FILES['Filedata']);
if ($_FILES['Filedata']['size'] <= $MAXIMUM_FILESIZE)
{
    move_uploaded_file($_FILES['Filedata']['tmp_name'],
"./temporary/".$_FILES['Filedata']['name']);
    $type = exif_imagetype("./temporary/".$_FILES['Filedata']['name']);
    if ($type == 1 || $type == 2 || $type == 3)
    {
        rename("./temporary/".$_FILES['Filedata']['name'],
"./images/".$_FILES['Filedata']['name']);
    }
    else
    {
        unlink("./temporary/".$_FILES['Filedata']['name']);
    }
}
$directory = opendir('./images/');
$files = array();
while ($file = readdir($directory))
{
    array_push($files, array('./images/'.$file, filectime('./images/'.$file)));
}
usort($files, sorter);
if (count($files) > $MAXIMUM_FILE_COUNT)
{
    $files_to_delete = array_splice($files, 0, count($files) - $MAXIMUM_FILE_COUNT);
    for ($i = 0; $i < count($files_to_delete); $i++)
    {
        unlink($files_to_delete[$i][0]);
    }
}
print_r($files);
closedir($directory);

function sorter($a, $b)
{
    if ($a[1] == $b[1])
    {
        return 0;
    }
    else
    {
        return ($a[1] < $b[1]) ? -1 : 1;
    }
}
?>
```

Mit der `POST`- oder `GET`-Anforderungsmethode können Sie zusätzliche Variablen an das Upload-Skript übergeben. Sie können den folgenden Code verwenden, um zusätzliche `POST`-Variablen an Ihr Upload-Skript zu übergeben:

```
var fileRef:FileReference = new FileReference();
fileRef.addEventListener(Event.SELECT, selectHandler);
fileRef.addEventListener(Event.COMPLETE, completeHandler);
fileRef.browse();
function selectHandler(event:Event):void
{
    var params:URLVariables = new URLVariables();
    params.date = new Date();
    params.ssid = "94103-1394-2345";
    var request:URLRequest = new
URLRequest("http://www.yourdomain.com/FileReferenceUpload/fileupload.cfm");
    request.method = URLRequestMethod.POST;
    request.data = params;
    fileRef.upload(request, "Custom1");
}
function completeHandler(event:Event):void
{
    trace("uploaded");
}
```

Das vorangegangene Codebeispiel erstellt ein URLVariables-Objekt, das Sie an das externe serverseitige Skript übergeben. In früheren Versionen von ActionScript konnten Sie Variablen durch Angabe von Werten im Abfrage-String an das Server-Upload-Skript übergeben. Mit ActionScript 3.0 können Sie Variablen mit einem URLRequest-Objekt an das externe Skript übergeben. Dies ermöglicht Ihnen die Übergabe von Daten mit der `POST`- oder `GET`-Methode, und das wiederum macht die Übergabe längerer Datensätze einfacher und übersichtlicher. Um anzugeben, welche Variablen mit der Anforderungsmethode `GET` oder `POST` übergeben werden, können Sie die Eigenschaft `URLRequest.method` auf `URLRequestMethod.GET` bzw. `URLRequestMethod.POST` einstellen.

ActionScript 3.0 ermöglicht Ihnen darüber hinaus das Überschreiben des standardmäßigen Namens für das `Filedata`-Upload-Dateifeld, indem Sie einen zweiten Parameter für die Methode `upload()` angeben. Dies wurde im vorangegangenen Codebeispiel demonstriert (in dem der Standardwert `Filedata` durch `Custom1` ersetzt wurde).

In der Standardeinstellung versucht Flash Player nicht, einen Test-Upload zu senden, obwohl Sie diese Standardeinstellung außer Kraft setzen können, indem Sie `true` als dritten Parameter für die `upload()`-Methode angeben. Der Zweck eines Test-Upload besteht darin, zu überprüfen, ob der tatsächliche Datei-Upload erfolgreich verlaufen wird und ob eine Authentifizierung beim Server (falls erforderlich) erfolgreich sein wird.

Hinweis: *Ein Test-Upload findet derzeit nur bei Windows-basierten Flash Player-Versionen statt.*

Das den Upload verwaltende Serverskript erwartet eine HTTP `POST`-Anforderung mit den folgenden Elementen:

- `Content-Type` mit einem Wert von `multipart/form-data`
- `Content-Disposition` mit einem Attribut `name`, das auf „`Filedata"` eingestellt ist, und mit einem Attribut `filename`, das auf den Namen der Originaldatei eingestellt ist. Sie können ein benutzerdefiniertes `name`-Attribut angeben, indem Sie einen Wert für den `uploadDataFieldName`-Parameter in der `FileReference.upload()`-Methode übergeben.
- Die binären Inhalte der Datei.

Im Folgenden finden Sie ein Beispiel einer HTTP `POST`-Anforderung:

```
POST /handler.asp HTTP/1.1
Accept: text/*
Content-Type: multipart/form-data;
boundary=----------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
User-Agent: Shockwave Flash
Host: www.mydomain.com
Content-Length: 421
Connection: Keep-Alive
Cache-Control: no-cache

------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
Content-Disposition: form-data; name="Filename"

sushi.jpg
------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
Content-Disposition: form-data; name="Filedata"; filename="sushi.jpg"
Content-Type: application/octet-stream

Test File
------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
Content-Disposition: form-data; name="Upload"

Submit Query
------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
(actual file data,,,)
```

Das folgende Beispiel einer HTTP `POST`-Anforderung sendet drei `POST`-Variablen: `api_sig`, `api_key` und `auth_token`. Weiterhin wird ein benutzerdefinierter Upload-Datenfeldname von „`photo"` verwendet:

```
POST /handler.asp HTTP/1.1
Accept: text/*
Content-Type: multipart/form-data;
boundary=----------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
User-Agent: Shockwave Flash
Host: www.mydomain.com
Content-Length: 421
Connection: Keep-Alive
Cache-Control: no-cache

------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="Filename"

sushi.jpg
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="api_sig"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="api_key"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="auth_token"

XXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="photo"; filename="sushi.jpg"
Content-Type: application/octet-stream

(actual file data,,,)
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="Upload"

Submit Query
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7--
```

Herunterladen von Dateien von einem Server

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Methode `FileReference.download()` können Sie Benutzern das Herunterladen von Dateien von einem Server ermöglichen. Dieser Methode lässt zwei Parametern zu: `request` und `defaultFileName`. Der erste Parameter ist ein URLRequest-Objekt, das die URL der herunterzuladenden Datei enthält. Der zweite Parameter ist optional. Mit ihm können Sie einen standardmäßigen Dateinamen angeben, der im Dialogfeld zum Herunterladen der Datei angezeigt wird. Wenn Sie den zweiten Parameter, `defaultFileName`, weglassen, wird der Dateiname aus der angegebenen URL verwendet.

Mit dem folgenden Code wird eine Datei namens „index.xml“ aus dem gleichen Verzeichnis wie die SWF-Datei heruntergeladen:

```
var request:URLRequest = new URLRequest("index.xml");
var fileRef:FileReference = new FileReference();
fileRef.download(request);
```

Um den Standardnamen von „index.xml“ in „currentnews.xml“ zu ändern, verwenden Sie den `defaultFileName`-Parameter. Dies wird im folgenden Codeausschnitt gezeigt:

```
var request:URLRequest = new URLRequest("index.xml");
var fileToDownload:FileReference = new FileReference();
fileToDownload.download(request, "currentnews.xml");
```

Das Umbenennen einer Datei ist insbesondere dann sinnvoll, wenn der Dateiname auf dem Server nicht besonders intuitiv ist oder vom Server erzeugt wurde. Außerdem sollten Sie den Parameter `defaultFileName` explizit angeben, wenn Sie eine Datei mit einem serverseitigen Skript und nicht direkt herunterladen. Beispielsweise müssen Sie den Parameter `defaultFileName` angeben, wenn Sie mit einem serverseitigen Skript arbeiten, das bestimmte Dateien basierend auf den an das Skript übergebenen URL-Variablen herunterlädt. Andernfalls wird der Name Ihres serverseitigem Skripts zum Standardnamen der heruntergeladenen Datei.

Daten können mit der `download()`-Methode an den Server übertragen werden, indem Sie der URL Parameter hinzufügen, die das Serverskript analysiert. Mit dem folgenden ActionScript 3.0-Codeausschnitt wird ein Dokument basierend auf den Parametern heruntergeladen, die an ein ColdFusion-Skript übergeben wurden:

```
package
{
    import flash.display.Sprite;
    import flash.net.FileReference;
    import flash.net.URLRequest;
    import flash.net.URLRequestMethod;
    import flash.net.URLVariables;

    public class DownloadFileExample extends Sprite
    {
        private var fileToDownload:FileReference;
        public function DownloadFileExample()
        {
            var request:URLRequest = new URLRequest();
            request.url = "http://www.[yourdomain].com/downloadfile.cfm";
            request.method = URLRequestMethod.GET;
            request.data = new URLVariables("id=2");
            fileToDownload = new FileReference();
            try
            {
                fileToDownload.download(request, "file2.txt");
            }
            catch (error:Error)
            {
                trace("Unable to download file.");
            }
        }
    }
}
```

Der folgende Code zeigt das ColdFusion-Skript „download.cfm“, das abhängig vom Wert der URL-Variablen eine von zwei Dateien vom Server herunterlädt:

```
<cfparam name="URL.id" default="1" />
<cfswitch expression="#URL.id#">
    <cfcase value="2">
        <cfcontent type="text/plain" file="#ExpandPath('two.txt')#" deletefile="No" />
    </cfcase>
    <cfdefaultcase>
        <cfcontent type="text/plain" file="#ExpandPath('one.txt')#" deletefile="No" />
    </cfdefaultcase>
</cfswitch>
```

FileReferenceList-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der FileReferenceList-Klasse können Benutzer eine oder mehrere Dateien zum Hochladen an ein serverseitiges Skript auswählen. Der Datei-Upload wird von der `FileReference.upload()`-Methode verarbeitet, die für jede vom Benutzer ausgewählte Datei aufgerufen werden muss.

Mit dem folgenden Code werden zwei FileFilter-Objekte (`imageFilter` und `textFilter`) erstellt und in einem Array an die `FileReferenceList.browse()`-Methode übergeben. Dies sorgt dafür, dass das Datei-Dialogfeld des Betriebssystems zwei mögliche Filter für Dateitypen anzeigt.

```
var imageFilter:FileFilter = new FileFilter("Image Files (*.jpg, *.jpeg, *.gif, *.png)",
"*.jpg; *.jpeg; *.gif; *.png");
var textFilter:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)", "*.txt; *.rtf");
var fileRefList:FileReferenceList = new FileReferenceList();
try
{
    var success:Boolean = fileRefList.browse(new Array(imageFilter, textFilter));
}
catch (error:Error)
{
    trace("Unable to browse for files.");
}
```

Das Auswählen und Hochladen einer oder mehrerer Dateien mithilfe der FileReferenceList-Klasse ist das Gleiche wie das Verwenden von `FileReference.browse()` zur Auswahl von Dateien, obwohl mit FileReferenceList mehrere Dateien gleichzeitig ausgewählt werden können. Beim Hochladen mehrerer Dateien müssen Sie jede ausgewählte Datei mit `FileReference.upload()` hochladen. Dies wird im folgenden Code gezeigt:

```
var fileRefList:FileReferenceList = new FileReferenceList();
fileRefList.addEventListener(Event.SELECT, selectHandler);
fileRefList.browse();

function selectHandler(event:Event):void
{
    var request:URLRequest = new URLRequest("http://www.[yourdomain].com/fileUploadScript.cfm");
    var file:FileReference;
    var files:FileReferenceList = FileReferenceList(event.target);
    var selectedFileArray:Array = files.fileList;
    for (var i:uint = 0; i < selectedFileArray.length; i++)
    {
        file = FileReference(selectedFileArray[i]);
        file.addEventListener(Event.COMPLETE, completeHandler);
        try
        {
            file.upload(request);
        }
        catch (error:Error)
        {
            trace("Unable to upload files.");
        }
    }
}
function completeHandler(event:Event):void
{
    trace("uploaded");
}
```

Da das Ereignis `Event.COMPLETE` jedem einzelnen FileReference-Objekt im Array hinzugefügt wird, ruft Flash Player die Methode `completeHandler()` auf, wenn das Hochladen für jede einzelne Datei abgeschlossen ist.

Verwenden der AIR-Dateisystem-API

Adobe AIR 1.0 und höher

Die Dateisystem-API von Adobe AIR enthält die folgenden Klassen:

- Datei
- FileMode
- FileStream

Mit der Dateisystem-API können Sie unter anderem Folgendes ausführen:

- Dateien und Verzeichnisse kopieren, erstellen, löschen und verschieben
- Informationen zu Dateien und Ordnern abrufen
- Dateien lesen und schreiben

Grundlegende Dateioperationen

Adobe AIR 1.0 und höher

Eine Kurzbeschreibung und Codebeispiele für die Arbeit mit dem Dateisystem in AIR finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Erstellen eines Textdatei-Editors (Flash)
- Building a text-file editor (Flex)
- Building a directory search application (Flex)
- Reading and writing from an XML preferences file (Flex)
- Compressing files and data (Flex)

Adobe AIR stellt Klassen bereit, mit denen Sie sowohl auf Dateien wie auch auf Ordner zugreifen können und mit denen Sie Dateien und Ordner erstellen und verwalten können. Diese Klassen, die sich im Paket „flash.filesystem“ befinden, haben folgende Funktion:

Dateiklassen	Beschreibung
File	Ein File-Objekt stellt einen Pfad zu einer Datei oder einem Verzeichnis dar. Mithilfe eines File-Objekts können Sie einen Zeiger auf eine Datei oder einen Ordner erstellen oder eine Interaktion mit einer Datei oder einem Ordner initiieren.
FileMode	Die FileMode-Klasse definiert Stringkonstanten, die im `fileMode`-Parameter der Methoden `open()` und `openAsync()` der FileStream-Klasse verwendet werden. Der `fileMode`-Parameter dieser Methoden bestimmt die Möglichkeiten des FileStream-Objekts, nachdem die Datei geöffnet wurde, namentlich Schreiben, Lesen, Anhängen und Aktualisieren.
FileStream	Ein FileStream-Objekt wird zum Öffnen von Dateien für das Lesen und Schreiben verwendet. Nachdem Sie ein File-Objekt erstellt haben, das auf eine neue oder vorhandene Datei zeigt, übergeben Sie diesen Zeiger an das FileStream-Objekt, sodass Sie die Datei öffnen und Daten lesen oder schreiben können.

Für manche Methoden in der File-Klasse gibt es eine synchrone wie auch eine asynchrone Version:

- `File.copyTo()` und `File.copyToAsync()`
- `File.deleteDirectory()` und `File.deleteDirectoryAsync()`
- `File.deleteFile()` und `File.deleteFileAsync()`
- `File.getDirectoryListing()` und `File.getDirectoryListingAsync()`
- `File.moveTo()` und `File.moveToAsync()`
- `File.moveToTrash()` und `File.moveToTrashAsync()`

Auch FileStream-Operationen arbeiten synchron oder asynchron, abhängig davon, wie das FileStream-Objekt die Datei öffnet: durch Aufruf der Methode `open()` oder der Methode `openAsync()`.

Die asynchronen Versionen erlauben die Initiierung von Prozessen, die im Hintergrund ablaufen und ein Ereignis auslösen, wenn sie abgeschlossen sind (oder wenn ein Fehler eintritt). Während diese asynchronen Hintergrundprozesse stattfinden, kann gleichzeitig anderer Code ausgeführt werden. Bei asynchronen Versionen der Operationen müssen Sie mithilfe der Methode `addEventListener()` des File- oder FileStream-Objekts, das die Funktion aufruft, eine Ereignis-Listener-Funktion einrichten.

Die synchronen Versionen ermöglichen es Ihnen, einfacheren Code zu schreiben, der nicht auf die Einrichtung von Ereignis-Listenern angewiesen ist. Da jedoch während der Ausführung einer synchronen Methode kein anderer Code ausgeführt werden kann, werden wichtige Prozesse wie das Rendern von Anzeigeobjekten oder eine Animation möglicherweise verzögert.

Arbeiten mit File-Objekten in AIR

Adobe AIR 1.0 und höher

Ein File-Objekt ist ein Zeiger auf eine Datei oder ein Verzeichnis im Dateisystem.

Die File-Klasse erweitert die FileReference-Klasse. Die FileReference-Klasse, die sowohl in Adobe® Flash® Player als auch in AIR verfügbar ist, repräsentiert einen Zeiger auf eine Datei. Die File-Klasse fügt Eigenschaften und Methoden hinzu, die aus Sicherheitsgründen in Flash Player (in einer SWF-Datei, die in einem Browser ausgeführt wird) nicht angezeigt werden.

File-Klasse

Adobe AIR 1.0 und höher

Mit der File-Klasse können Sie folgende Aktionen ausführen:

- Abrufen des Pfads zu speziellen Verzeichnissen, einschließlich des Benutzerverzeichnisses, des Dokumentverzeichnisses des Benutzers, des Verzeichnisses, aus dem die Anwendung gestartet wurde, und des Anwendungsverzeichnisses.
- Kopieren von Dateien und Verzeichnissen
- Verschieben von Dateien und Verzeichnissen
- Löschen von Dateien und Verzeichnissen (bzw. Verschieben in den Papierkorb)
- Auflisten von Dateien und Verzeichnissen in einem bestimmten Verzeichnis
- Erstellen temporärer Dateien und Ordner

Sobald ein File-Objekt auf einen Dateipfad zeigt, können Sie mithilfe der FileStream-Klasse Daten in der Datei lesen oder Daten in die Datei schreiben.

Ein File-Objekt kann auch auf den Pfad einer Datei oder eines Verzeichnisses zeigen, das noch gar nicht existiert. Sie können ein solches File-Objekt beim Erstellen einer Datei oder eines Verzeichnisses verwenden.

Pfade von File-Objekten

Adobe AIR 1.0 und höher

Jedes File-Objekt hat zwei Eigenschaften, die jeweils seinen Pfad definieren:

Eigenschaft	Beschreibung
`nativePath`	Gibt den plattformspezifischen Pfad zu einer Datei an. Unter Windows könnte ein Pfad beispielsweise „c:\Beispielverzeichnis\test.txt“ lauten und unter Mac OS „/Beispielverzeichnis/test.txt“. Eine `nativePath`-Eigenschaft verwendet den umgekehrten Schrägstrich (\) als Verzeichnistrenner unter Windows und den vorwärts gerichteten Schrägstrich (/) unter Mac OS und Linux.
`url`	Hier kann das file-URL-Schema verwendet werden, um auf eine Datei zu zeigen. Unter Windows könnte ein Pfad beispielsweise „file:///c:/Beispielverzeichnis/test.txt“ lauten und unter Mac OS „file:///Beispielverzeichnis/test.txt". Die Laufzeitumgebung umfasst neben dem `file`-Schema weitere spezielle URL-Schemas. Diese werden unter „Unterstützte AIR-URL-Schemata“ auf Seite 721 erläutert.

Die File-Klasse enthält statische Eigenschaften, um auf Standardverzeichnisse von MAC OS, Windows und Linux zu zeigen. Zu diesen Eigenschaften gehören:

- `File.applicationStorageDirectory`: ein für jede der installierten AIR-Anwendungen eindeutiger Speicherordner. Dieses Verzeichnis eignet sich zum Speichern von dynamischen Anwendungselementen und von Benutzervoreinstellungen. Große Datenmengen sollten in einem anderen Verzeichnis gespeichert werden.

  Auf Android-Geräten und unter iOS wird das Anwendungsspeicherverzeichnis entfernt, wenn der Benutzer die Anwendung deinstalliert oder Anwendungsdaten löscht; auf anderen Plattformen ist dies jedoch nicht der Fall.

- `File.applicationDirectory` – das Verzeichnis, in dem die Anwendung installiert wird (zusammen mit ggf. installierten Beständen). Unter einigen Betriebssystemen wird die Anwendung in einer einzelnen Paketdatei gespeichert, nicht in einem physischen Verzeichnis. In diesem Fall ist der Zugriff auf den Inhalt über den nativen Pfad eventuell nicht möglich. Das Anwendungsverzeichnis ist schreibgeschützt.
- `File.desktopDirectory` – das Desktopverzeichnis des Benutzers. Wenn auf der Plattform kein Desktopverzeichnis definiert ist, wird ein anderes Verzeichnis im Dateisystem verwendet.
- `File.documentsDirectory` – das Dokumentverzeichnis des Benutzers. Wenn auf der Plattform kein Dokumentverzeichnis definiert ist, wird ein anderes Verzeichnis im Dateisystem verwendet.
- `File.userDirectory`: das Benutzerverzeichnis. Wenn auf der Plattform kein Benutzerverzeichnis definiert ist, wird ein anderes Verzeichnis im Dateisystem verwendet.

Hinweis: *Wenn auf der Plattform keine Standardorte für das Desktop-, Dokument- oder Benutzerverzeichnis definiert sind, können* `File.documentsDirectory`, `File.desktopDirectory` *und* `File.userDirectory` *auf dasselbe Verzeichnis verweisen.*

Diese Eigenschaften verfügen auf verschiedenen Betriebssystemen über unterschiedliche Werte. Zum Beispiel verwenden Mac und Windows jeweils unterschiedliche native Pfade zum Desktopverzeichnis des Benutzers. Die `File.desktopDirectory`-Eigenschaft verweist jedoch auf jeder Plattform auf einen geeigneten Verzeichnispfad. Nutzen Sie diese Eigenschaften als Grundlage für Verweise auf andere Verzeichnisse und Dateien, die von der Anwendung verwendet werden, um Anwendungen zu schreiben, die auf allen Plattformen gleichermaßen gut funktionieren. Verwenden Sie anschließend die `resolvePath()`-Methode für die genauere Festlegung des Pfads. Der nachfolgende Code verweist z. B. auf die Datei „preferences.xml“ im Speicherordner der Anwendung.

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("preferences.xml");
```

Mit der File-Klasse können Sie zwar auf einen spezifischen Dateipfad zeigen, damit erhalten Sie möglicherweise jedoch Anwendungen, die nicht auf allen Plattformen funktionieren. Der Pfad C:\Dokumente und Einstellungen\joe\ funktioniert zum Beispiel nur unter Windows. Deshalb sind die statischen Eigenschaften der File-Klasse besser geeignet, zum Beispiel `File.documentsDirectory`.

Gemeinsame Verzeichnispfade

Plattform	Verzeichnistyp	Typischer Pfad im Dateisystem
Android	Anwendung	`/data/data/`
	Anwendungsspeicher	`/data/data/air.Anwendungs_ID/Dateiname/Local Store`
	Cache	/data/data/Anwendungs_ID/cache
	Desktop	`/mnt/sdcard`
	Dokumente	`/mnt/sdcard`
	Temporär	`/data/data/Anwendungs_ID/cache/FlashTmp.randomString`
	Benutzer	`/mnt/sdcard`
iOS	Anwendung	`/var/mobile/Applications/uid/filename.app`
	Anwendungsspeicher	`/var/mobile/Applications/uid/Library/Application Support/applicationID/Local Store`
	Cache	/var/mobile/Applications/uid/Library/Caches
	Desktop	nicht zugriffsbereit
	Dokumente	`/var/mobile/Applications/uid/Documents`
	Temporär	`/private/var/mobile/Applications/uid/tmp/FlashTmpNNN`
	Benutzer	nicht zugriffsbereit
Linux	Anwendung	`/opt/Dateiname/share`
	Anwendungsspeicher	`/home/Benutzername/.appdata/Anwendungs_ID/Local Store`
	Desktop	`/home/Benutzername/Desktop`
	Dokumente	`/home/Benutzername/Documents`
	Temporär	`/tmp/FlashTmp.randomString`
	Benutzer	`/home/Benutzername`
Mac	Anwendung	`/Applications/filename.app/Contents/Resources`
	Anwendungsspeicher	`/Benutzer/`***`Benutzername`***`/Library/Preferences/`***`applicationid`***`/Local Store` (AIR 3.2 und früher) ***`Pfad`***`/Library/Application Support/`***`applicationid`***`Local Store` (AIR 3.3 und neuer), wobei „Pfad“ entweder `/Benutzer/`***`Benutzername`***`/Library/Containers/`***`bundle-id`***`/Data` (Sandbox-Umgebung) oder `/Benutzer/Benutzername` (bei Ausführung außerhalb einer Sandbox-Umgebung) ist
	Cache	/Benutzer/Benutzername/Library/Caches
	Desktop	`/Benutzer/`***`Benutzername`***`/Desktop`
	Dokumente	`/Benutzer/`***`Benutzername`***`/Dokumente`
	Temporär	`/private/var/folders/JY/randomString/TemporaryItems/FlashTmp`
	Benutzer	`/Benutzer/`***`Benutzername`***

Plattform	Verzeichnistyp	Typischer Pfad im Dateisystem
Windows	Anwendung	`C:\Programme\Dateiname`
	Anwendungsspeicher	`C:\Dokumente und Einstellungen\Benutzername\Anwendungsdaten\Anwendungs_ID\Local Store`
	Cache	C:\Dokumente und Einstellungen\Benutzername\Lokale Einstellungen\Temp
	Desktop	`C:\Dokumente und Einstellungen\Benutzername\Desktop`
	Dokumente	`C:\Dokumente und Einstellungen\Benutzername\Eigene Dokumente`
	Temporär	`C:\Dokumente und Einstellungen\Benutzername\Lokale Einstellungen\Temp\randomString.tmp`
	Benutzer	`C:\Dokumente und Einstellungen\Benutzername`

Die tatsächlichen nativen Pfade dieser Verzeichnisse variieren je nach Betriebssystem und Computerkonfiguration. Die in dieser Tabelle genannten Pfade sind typische Beispiele. Sie sollten stets mit den entsprechenden Eigenschaften der statischen File-Klasse auf diese Verzeichnisse verweisen, um sicherzustellen, dass Ihre Anwendung auf jeder Plattform korrekt funktioniert. In einer echten AIR-Anwendung stammen die in der Tabelle gezeigten Werte für `Anwendungs_ID` und `Dateiname` aus dem Anwendungsdeskriptor. Wenn Sie im Anwendungsdeskriptor eine Herausgeber-ID angeben, wird diese ID in diesen Pfaden an die Anwendungs-ID angehängt. Der Wert für `Benutzername` ist der Kontoname des Benutzers, der die Installation durchführt.

Verweisen vom File-Objekt auf ein Verzeichnis

Adobe AIR 1.0 und höher

Es gibt verschiedene Möglichkeiten, um ein File-Objekt so einzurichten, dass es auf ein Verzeichnis zeigt.

Verweisen auf das Stammverzeichnis eines Benutzers

Adobe AIR 1.0 und höher

Sie können einen Zeiger vom File-Objekt zum Stammverzeichnis definieren. Im folgenden Beispiel wird ein File-Objekt so eingerichtet, dass es auf das Unterverzeichnis „AIR Test“ des Stammverzeichnisses zeigt:

```
var file:File = File.userDirectory.resolvePath("AIR Test");
```

Verweisen auf das Dokumentenverzeichnis eines Benutzers

Adobe AIR 1.0 und höher

Sie können einen Verweis vom File-Objekt auf das Dokumentverzeichnis eines Benutzers definieren. Im folgenden Beispiel wird ein File-Objekt so eingerichtet, dass es auf das Unterverzeichnis „AIR Test“ des Dokumentverzeichnisses zeigt:

```
var file:File = File.documentsDirectory.resolvePath("AIR Test");
```

Verweisen auf das Desktopverzeichnis eines Benutzers

Adobe AIR 1.0 und höher

Sie können einen Zeiger vom File-Objekt zum Desktopverzeichnis eines Benutzers definieren. Im folgenden Beispiel wird ein File-Objekt so eingerichtet, dass es auf das Unterverzeichnis „AIR Test“ des Desktopverzeichnisses zeigt:

```
var file:File = File.desktopDirectory.resolvePath("AIR Test");
```

Verweisen auf das Anwendungsspeicherverzeichnis

Adobe AIR 1.0 und höher

Sie können einen Zeiger vom File-Objekt zum Anwendungsspeicherverzeichnis definieren. Für jede AIR-Anwendung gibt es einen eindeutigen verknüpften Pfad, der das Anwendungsspeicherverzeichnis definiert. Dieses Verzeichnis ist für jede Anwendung und jeden Benutzer eindeutig. Dieses Verzeichnis eignet sich zum Speichern benutzerspezifischer, anwendungsspezifischer Daten (z. B. Benutzerdaten oder eine Datei mit Einstellungen). Der folgende Code etwa verweist von einem File-Objekt auf die Einstellungsdatei „prefs.xml“, die sich im Anwendungsspeicherverzeichnis befindet.

```
var file:File = File.applicationStorageDirectory;
file = file.resolvePath("prefs.xml");
```

Das Anwendungsspeicherverzeichnis basiert normalerweise auf dem Benutzernamen und der Anwendungs-ID. Die folgenden Dateisystemverzeichnisse sollen Ihnen beim Debuggen Ihrer Anwendung helfen. Verwenden Sie immer die `File.applicationStorage`-Eigenschaft oder das `app-storage:`-URI-Schema zum Auflösen der Dateien in diesem Verzeichnis:

- Unter Mac OS – abhängig von der AIR-Version:

  AIR 3.2 und älter: `/Benutzer/Benutzername/Library/Preferences/`*`applicationID`*`/Local Store/`

  AIR 3.3 und höher: *`Pfad`*`/Library/Application Support/`*`applicationID`*`/Local Store`, wobei *`Pfad`* entweder `/Benutzer/`*`Benutzername`*`/Library/Containers/`*`bundle-id`*`/Data` (Sandbox-Umgebung) oder `/Benutzer/`*`Benutzername`* ist (bei Ausführung außerhalb einer Sandbox-Umgebung)

  Beispiel (AIR 3.2):

  ```
  /Users/babbage/Library/Preferences/com.example.TestApp/Local Store
  ```

- Unter Windows – im Verzeichnis „Dokumente und Einstellungen“:

  C:\Dokumente und Einstellungen\Benutzername`\Anwendungsdaten\`*Anwendungs_ID*`\Local Store\`

  Zum Beispiel:

  ```
  C:\Documents and Settings\babbage\Application Data\com.example.TestApp\Local Store
  ```

- Unter Linux – in:

  `/home/`*Benutzername*`/.appdata/`*Anwendungs_ID*`/Local Store/`

  Zum Beispiel:

  ```
  /home/babbage/.appdata/com.example.TestApp/Local Store
  ```

- Unter Android – in:

  /data/data/`android_Paket_ID`/`Anwendungs-ID/Local Store`

  Zum Beispiel:

  ```
  /data/data/air.com.example.TestApp/com.example.TestApp/Local Store
  ```

Hinweis: *Wenn eine Anwendung über eine Herausgeber-ID verfügt, gehört diese ebenfalls zum Pfad des Anwendungsspeicherverzeichnisses.*

Die URL (und die `url`-Eigenschaft) eines File-Objekts, das mit `File.applicationStorageDirectory` erstellt wurde, verwendet das URL-Schema `app-storage` (siehe „Unterstützte AIR-URL-Schemata“ auf Seite 721). Hier ein Codebeispiel:

```
var dir:File = File.applicationStorageDirectory;
dir = dir.resolvePath("preferences");
trace(dir.url); // app-storage:/preferences
```

Verweisen auf das Anwendungsverzeichnis

Adobe AIR 1.0 und höher

Sie können einen Zeiger vom File-Objekt zu dem Verzeichnis, in dem die Anwendung installiert worden ist, dem so genannten Anwendungsverzeichnis, definieren. Sie können mithilfe der Eigenschaft `File.applicationDirectory` auf dieses Verzeichnis verweisen. Sie können dieses Verzeichnis verwenden, um die Anwendungsdeskriptordatei oder andere mit der Anwendung installierten Ressourcen zu untersuchen. Der folgende Code etwa verweist von einem File-Objekt auf ein Verzeichnis im Anwendungsverzeichnis mit dem Namen *images*:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
```

Die URL (und die `url`-Eigenschaft) eines File-Objekts, das mit `File.applicationDirectory` erstellt wurde, verwendet das URL-Schema `app` (siehe „Unterstützte AIR-URL-Schemata“ auf Seite 721). Hier ein Codebeispiel:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
trace(dir.url); // app:/images
```

Hinweis: *Bei Android ist der Zugriff auf die Dateien im Anwendungspaket nicht über `nativePath` möglich. Die `nativePath`-Eigenschaft ist ein leerer String. Verwenden Sie für den Zugriff auf die Dateien im Anwendungsverzeichnis immer die URL anstelle des nativen Pfads.*

Verweisen auf das Cacheverzeichnis

Adobe AIR 3.6 und höher

Sie können ein File-Objekt mithilfe der Eigenschaft `File.cacheDirectory` auf das temporäre Verzeichnis oder Cacheverzeichnis des Betriebssystems verweisen. Dieses Verzeichnis enthält temporäre Dateien, die nicht zum Ausführen der Anwendung erforderlich sind und nicht zu Problemen bzw. Datenverlust führen, falls sie gelöscht werden.

Unter den meisten Betriebssystemen ist das Cacheverzeichnis ein temporäres Verzeichnis. Unter iOS entspricht das Cacheverzeichnis dem Cacheverzeichnis der Anwendungsbibliothek. Dateien in diesem Verzeichnis werden nicht im Online-Speicher gesichert und möglicherweise vom Betriebssystem gelöscht, falls das Gerät nicht über ausreichend freien Speicherplatz verfügt. Weitere Informationen finden Sie unter „Steuern der Dateisicherung und -zwischenspeicherung“ auf Seite 722.

Verweisen auf das Stammverzeichnis des Dateisystems

Adobe AIR 1.0 und höher

Die `File.getRootDirectories()`-Methode gibt eine Liste aller Stammspeichermedien auf einem Windows-Computer aus, z. B. „C:“ oder auch gemountete Speichermedien. Unter Mac OS und Linux liefert diese Methode immer das eindeutige Stammverzeichnis des Rechners, nämlich "/". Die `StorageVolumeInfo.getStorageVolumes()`-Methode liefert ausführlichere Informationen über gemountete Speichermedien (siehe „Arbeiten mit Speichermedien“ auf Seite 733).

Hinweis: *Das Stammverzeichnis des Dateisystems kann unter Android nicht gelesen werden. Ein File-Objekt, das mit dem nativen Pfad „/“ auf das Verzeichnis verweist, wird zurückgegeben, aber die Werte der Eigenschaften dieses Objekts sind ungenau. Beispielsweise ist `spaceAvailable` immer 0.*

Verweisen auf ein bestimmtes Verzeichnis

Adobe AIR 1.0 und höher

Sie können vom File-Objekt auf ein bestimmtes Verzeichnis verweisen, indem Sie die `nativePath`-Eigenschaft des File-Objekts setzen. Hier ein Beispiel unter Windows:

```
var file:File = new File();
file.nativePath = "C:\\AIR Test";
```

Wichtig: Wenn Sie auf diese Weise auf einen expliziten Pfad verweisen, funktioniert der Code möglicherweise nicht auf allen Plattformen. Das oben genannte Beispiel funktioniert beispielsweise nur unter Windows. Verwenden Sie die statischen Eigenschaften des File-Objekts, zum Beispiel `File.applicationStorageDirectory`, um so auf ein Verzeichnis zu zeigen, dass die Anwendung auf allen Plattformen funktioniert. Mit der `resolvePath()`-Methode (siehe nächster Abschnitt) können Sie dann zu einem relativen Pfad navigieren.

Navigieren zu relativen Verzeichnispfaden

Adobe AIR 1.0 und höher

Mit der Methode `resolvePath()` können Sie einen Pfad in Relation zu einem anderen, gegebenen Pfad abrufen. Im folgenden Code beispielsweise wird ein File-Objekt so eingerichtet, dass es auf das Unterverzeichnis „AIR Test“ des Stammverzeichnisses eines Benutzers zeigt:

```
var file:File = File.userDirectory;
file = file.resolvePath("AIR Test");
```

Sie können auch die `url`-Eigenschaft eines File-Objekts verwenden, um anhand eines URL-Strings auf ein Verzeichnis zu zeigen:

```
var urlStr:String = "file:///C:/AIR Test/";
var file:File = new File()
file.url = urlStr;
```

Weitere Informationen hierzu finden Sie unter „Ändern von Dateipfaden“ auf Seite 721.

Verzeichnisauswahl über die Verzeichnisstruktur

Adobe AIR 1.0 und höher

Die File-Klasse beinhaltet die Methode `browseForDirectory()`, die ein Systemdialogfeld anzeigt, in dem der Benutzer die Verzeichnisstruktur durchsuchen kann, um ein Verzeichnis auszuwählen und es dem Objekt zuzuweisen. `browseForDirectory()` ist eine asynchrone Methode. Das File-Objekt löst ein `select`-Ereignis aus, wenn der Benutzer ein Verzeichnis auswählt und auf die Schaltfläche „Öffnen“ klickt. Oder sie löst ein `cancel`-Ereignis aus, wenn der Benutzer auf die Schaltfläche „Abbrechen“ klickt.

Der folgende Code etwa erlaubt dem Benutzer die Auswahl eines Verzeichnisses aus der Verzeichnisstruktur und gibt dann den Verzeichnispfad aus:

```
var file:File = new File();
file.addEventListener(Event.SELECT, dirSelected);
file.browseForDirectory("Select a directory");
function dirSelected(e:Event):void {
    trace(file.nativePath);
}
```

Hinweis: *Unter Android wird die `browseForDirectory()`-Methode nicht unterstützt. Ein Aufruf dieser Methode hat keinerlei Auswirkung, da sofort ein cancel-Ereignis ausgelöst wird. Um Benutzern die Möglichkeit zu geben, ein Verzeichnis auszuwählen, stellen Sie stattdessen ein benutzerdefiniertes Dialogfeld für die Anwendung bereit.*

Verweisen auf das Verzeichnis, von dem aus die Anwendung aufgerufen wurde

Adobe AIR 1.0 und höher

Sie können den Pfad eines Verzeichnisses abrufen, von dem aus die Anwendung aufgerufen wurde. Dazu prüfen Sie die `currentDirectory`-Eigenschaft des InvokeEvent-Objekts, das beim Aufrufen der Anwendung ausgelöst worden ist. Weitere Informationen finden Sie unter „Erfassen von Befehlszeilenargumenten“ auf Seite 937.

Verweisen vom File-Objekt zu einer Datei

Adobe AIR 1.0 und höher

Es gibt verschiedene Möglichkeiten, um die Datei festzulegen, auf die ein File-Objekt zeigt.

Verweisen auf einen bestimmten Verzeichnispfad

Adobe AIR 1.0 und höher

Wichtig: Wenn Sie auf einen expliziten Pfad verweisen, funktioniert der Code möglicherweise nicht auf allen Plattformen. Der Pfad „C:/foo.txt“ funktioniert zum Beispiel nur unter Windows. Verwenden Sie die statischen Eigenschaften des File-Objekts, zum Beispiel `File.applicationStorageDirectory`, um so auf ein Verzeichnis zu zeigen, dass die Anwendung auf allen Plattformen funktioniert. Mit der `resolvePath()`-Methode (siehe „Ändern von Dateipfaden“ auf Seite 721) können Sie dann zu einem relativen Pfad navigieren.

Sie können die `url`-Eigenschaft eines File-Objekts verwenden, um anhand eines URL-Strings auf eine Datei oder ein Verzeichnis zu zeigen:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File()
file.url = urlStr;
```

Sie können den URL auch wie folgt an die Konstruktorfunktion `File()` übergeben:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File(urlStr);
```

Die Eigenschaft `url` gibt immer die URI-kodierte Version des URLs zurück (wobei Leerzeichen beispielsweise durch `"%20` ersetzt werden):

```
file.url = "file:///c:/AIR Test";
trace(file.url); // file:///c:/AIR%20Test
```

Sie können auch die `nativePath`-Eigenschaft eines File-Objekts verwenden, um einen bestimmten Pfad zu setzen. Wird der folgende Code beispielsweise auf einem Windows-Computer ausgeführt, wird ein File-Objekt auf die Datei „test.txt“ im Unterverzeichnis „AIR Test“ des Laufwerks „C:“ gesetzt:

```
var file:File = new File();
file.nativePath = "C:/AIR Test/test.txt";
```

Sie können diesen Pfad auch wie folgt an die Konstruktorfunktion `File()` übergeben:

```
var file:File = new File("C:/AIR Test/test.txt");
```

Verwenden Sie den Schrägstrich (/) als Pfadtrenner für die `nativePath`-Eigenschaft. Unter Windows können Sie auch den umgekehrten Schrägstrich (\) verwenden, dies führt jedoch zu Anwendungen, die nicht auf allen Plattformen funktionieren.

Weitere Informationen hierzu finden Sie unter „Ändern von Dateipfaden“ auf Seite 721.

Aufzählen von Dateien in einem Verzeichnis

Adobe AIR 1.0 und höher

Mithilfe der `getDirectoryListing()`-Methode eines File-Objekts können Sie einen Array mit File-Objekten abrufen, die auf Dateien und Unterverzeichnisse auf der Stammebene eines Verzeichnisses zeigen. Weitere Informationen finden Sie unter „Erstellen von Verzeichnislisten“ auf Seite 728.

Dateiauswahl über die Verzeichnisstruktur

Adobe AIR 1.0 und höher

Die File-Klasse beinhaltet die folgenden Methoden, die ein Systemdialogfeld anzeigen, in dem der Benutzer die Verzeichnisstruktur durchsuchen kann, um eine Datei auszuwählen und sie dem Objekt zuzuweisen.

- `browseForOpen()`
- `browseForSave()`
- `browseForOpenMultiple()`

Diese Methoden sind alle asynchron. Die Methoden `browseForOpen()` und `browseForSave()` lösen das Ereignis „select“ aus, wenn der Benutzer eine Datei auswählt (bzw. im Falle von „browseForSave()“ einen Zielpfad). Bei den Methoden `browseForOpen()` und `browseForSave()` zeigt das File-Objekt bei einer Auswahl auf die ausgewählten Dateien. Die Methode `browseForOpenMultiple()` löst ein `selectMultiple`-Ereignis aus, wenn der Benutzer mehrere Dateien auswählt. Das Ereignis `selectMultiple` hat den Typ „FileListEvent“, der eine `files`-Eigenschaft besitzt, die ein Array mit File-Objekten darstellt (die auf die ausgewählten Dateien zeigen).

Im folgenden Codebeispiel wird dem Benutzer ein Open-Dialogfeld anzeigt, in dem der Benutzer eine Datei auswählen kann:

```
var fileToOpen:File = File.documentsDirectory;
selectTextFile(fileToOpen);

function selectTextFile(root:File):void
{
    var txtFilter:FileFilter = new FileFilter("Text", "*.as;*.css;*.html;*.txt;*.xml");
    root.browseForOpen("Open", [txtFilter]);
    root.addEventListener(Event.SELECT, fileSelected);
}

function fileSelected(event:Event):void
{
    trace(fileToOpen.nativePath);
}
```

Ist in der Anwendung ein anderes Durchsuchen-Dialogfeld geöffnet, wenn Sie eine browse-Methode aufrufen, wird von der Laufzeitumgebung eine Error-Ausnahme ausgelöst.

Hinweis: *Unter Android können nur Bild-, Video- und Audiodateien mit den Methoden* `browseForOpen()` *und* `browseForOpenMultiple()` *ausgewählt werden. Das mit browseForSave() aufgerufene Dialogfeld zeigt ebenfalls nur Mediendateien an, obwohl der Benutzer einen beliebigen Dateinamen eingeben kann. Zum Öffnen und Speichern von Dateien, bei denen es sich nicht um Medien handelt, empfiehlt sich die Verwendung von benutzerdefinierten Dialogfeldern anstelle dieser Methoden.*

Ändern von Dateipfaden

Adobe AIR 1.0 und höher

Sie können auch den Pfad eines vorhandenen File-Objekts ändern, indem Sie die Methode `resolvePath()` aufrufen oder die Eigenschaft `nativePath` oder `url` des Objekts ändern. Hier ein Beispiel unter Windows:

```
var file1:File = File.documentsDirectory;
file1 = file1.resolvePath("AIR Test");
trace(file1.nativePath); // C:\Documents and Settings\userName\My Documents\AIR Test
var file2:File = File.documentsDirectory;
file2 = file2.resolvePath("..");
trace(file2.nativePath); // C:\Documents and Settings\userName
var file3:File = File.documentsDirectory;
file3.nativePath += "/subdirectory";
trace(file3.nativePath); // C:\Documents and Settings\userName\My Documents\subdirectory
var file4:File = new File();
file4.url = "file:///c:/AIR Test/test.txt";
trace(file4.nativePath); // C:\AIR Test\test.txt
```

Wenn Sie mit der `nativePath`-Eigenschaft arbeiten, verwenden Sie den Schrägstrich (/) Verzeichnistrennzeichen. Unter Windows können Sie auch den umgekehrten Schrägstrich (\) verwenden, dies sollte jedoch vermieden werden, da der entsprechende Code dann nicht auf allen Plattformen verwendet werden kann.

Unterstützte AIR-URL-Schemata

Adobe AIR 1.0 und höher

In AIR können Sie beim Definieren der `url`-Eigenschaft eines File-Objekts eines der folgenden URL-Schemata verwenden:

URL-Schema	Beschreibung
file	Verwenden Sie dieses Schema, um einen Pfad in Relation zum Stammverzeichnis des Dateisystems anzugeben. Zum Beispiel: `file:///c:/AIR Test/test.txt` Der URL-Standard legt für eine Datei-URL das Format `file://<Host>/<Pfad>` fest. Ein möglicher Sonderfall wäre, dass `<Host>` ein leerer String ist, der als „der Rechner, von dem die URL interpretiert wird" umgesetzt wird. Daher haben die URLs oft auch drei Schrägstriche (///).
app	Verwenden Sie dieses Schema, um einen Pfad in Relation zum Stammverzeichnis der installierten Anwendung anzugeben (der Anwendung, die die Datei „application.xml" für die installierte Anwendung enthält). Der folgende Pfad zeigt zum Beispiel auf ein images-Unterverzeichnis der installierten Anwendung. `app:/images`
app-storage	Verwenden Sie dieses Schema, um einen Pfad in Relation zum Anwendungsspeicherverzeichnis anzugeben. AIR definiert für jede installierte Anwendung ein Anwendungsspeicherverzeichnis, indem man anwendungsspezifische Daten speichern kann. Der folgende Pfad etwa verweist von einem File-Objekt auf eine Datei mit dem Namen „prefs.xml" in einem settings-Verzeichnis des Anwendungsspeicherverzeichnisses: `app-storage:/settings/prefs.xml`

Steuern der Dateisicherung und -zwischenspeicherung

Adobe AIR 3.6 und höher – nur iOS und OS X

Unter bestimmten Betriebssystemen, insbesondere iOS und Mac OS X, können Benutzer Anwendungsdateien automatisch in einem Remote-Speicher sichern. Unter iOS unterliegt es gewissen Einschränkungen, ob Dateien gesichert werden können und an welchem Ort Dateien für verschiedene Zwecke gespeichert werden können.

Die folgenden Punkte fassen zusammen, wie Sie die Richtlinien von Apple für die Dateisicherung und -speicherung einhalten. Weitere Informationen finden Sie in den nächsten Abschnitten.

- Um anzugeben, dass eine Datei nicht gesichert werden muss und (nur iOS) vom Betriebssystem gelöscht werden kann, falls der Speicherplatz des Geräts nicht mehr ausreicht, speichern Sie die Datei im Cacheverzeichnis (`File.cacheDirectory`). Dies ist der bevorzugte Speicherort unter iOS und sollte für die meisten Dateien verwendet werden, die neu generiert oder heruntergeladen werden können.
- Um anzugeben, dass eine Datei nicht gesichert werden muss, jedoch nicht vom Betriebssystem gelöscht werden soll, speichern Sie die Datei in einem der Verzeichnisse der Anwendungsbibliothek, z. B. im Anwendungsspeicherverzeichnis (`File.applicationStorageDirectory`) oder im Dokumentenverzeichnis (`File.documentsDirectory`). Setzen Sie die Eigenschaft `preventBackup` des File-Objekts auf `true`. Dies ist laut Apple erforderlich bei Inhalten, die neu generiert oder heruntergeladen werden können, jedoch für das einwandfreie Funktionieren Ihrer Anwendung bei Offline-Verwendung notwendig sind.

Angeben von Dateien für die Sicherung

Um Speicherplatz für die Datensicherung zu sparen und die Netzwerkbandbreite zu minimieren, geben die Richtlinien von Apple für iOS- und Mac-Anwendungen an, dass nur Dateien mit benutzerdefinierten Daten bzw. mit Daten, die nicht neu generiert oder herunterladen werden können, für die Sicherung vorgesehen werden sollen.

Standardmäßig werden alle Dateien in den Bibliotheksordnern der Anwendung gesichert. Unter Mac OS X ist dies das Anwendungsspeicherverzeichnis. Unter iOS umfasst dies das Anwendungsspeicherverzeichnis, das Anwendungsverzeichnis, das Desktopverzeichnis, das Dokumentenverzeichnis sowie das Benutzerverzeichnis (da diese Verzeichnisse den Bibliotheksordnern der Anwendung unter iOS zugeordnet sind). Somit werden alle Dateien in diesen Verzeichnissen standardmäßig im Serverspeicher gesichert.

Wenn Sie eine Datei, die von Ihrer Anwendung neu erstellt werden kann, an einen dieser Speicherorte speichern, markieren Sie die Datei, sodass das Betriebssystem sie nicht sichert. Um anzugeben, dass eine Datei nicht gesichert werden soll, setzen Sie die Eigenschaft `preventBackup` des File-Objekts auf `true`.

Unter iOS werden Dateien in einem der Bibliotheksordner der Anwendung als permanente Dateien markiert, die nicht vom Betriebssystem gelöscht werden sollen, selbst wenn die Eigenschaft `preventBackup` der Datei auf `true` gesetzt ist.

Steuern der Dateizwischenspeicherung und -löschung

Die Richtlinien von Apple für iOS legen fest, dass soweit möglich Inhalte, die neu generiert werden können, vom Betriebssystem gelöscht werden sollen, wenn der Speicherplatz nicht ausreicht.

Unter iOS werden Dateien in den Bibliotheksordnern der Anwendung (z. B. Anwendungsspeicherverzeichnis oder Dokumentenverzeichnis) als permanent markiert und vom Betriebssystem nicht gelöscht.

Speichern Sie Dateien, die von der Anwendung neu generiert und bei nicht ausreichendem Speicherplatz im Cacheverzeichnis der Anwendung gelöscht werden können. Greifen Sie mithilfe der statischen Eigenschaft `File.cacheDirectory` auf das Cacheverzeichnis zu.

Unter iOS entspricht das Cacheverzeichnis dem Cacheverzeichnis der Anwendung (<Application Home>/Library/Caches). Unter anderen Betriebssystemen ist dieses Verzeichnis einem vergleichbaren Verzeichnis zugeordnet. Unter Mac OS X ist es auch dem Cacheverzeichnis in der Anwendungsbibliothek zugeordnet. Unter Android ist das Cacheverzeichnis dem Cacheverzeichnis der Anwendung zugeordnet. Unter Windows ist das Cacheverzeichnis dem temporären Verzeichnis des Betriebssystems zugeordnet. Sowohl unter Android als auch unter Windows ist dies das Verzeichnis, auf das durch den Aufruf der `createTempDirectory()`- und `createTempFile()`-Methoden der File-Klasse zugegriffen wird.

Suchen nach dem relativen Pfad zwischen zwei Dateien

Adobe AIR 1.0 und höher

Mit der Methode `getRelativePath()` können Sie den relativen Pfad zwischen zwei Dateien ermitteln:

```
var file1:File = File.documentsDirectory.resolvePath("AIR Test");
var file2:File = File.documentsDirectory
file2 = file2.resolvePath("AIR Test/bob/test.txt");

trace(file1.getRelativePath(file2)); // bob/test.txt
```

Der zweite Parameter der Methode `getRelativePath()`, der Parameter `useDotDot` bewirkt, dass in den Ergebnissen die `..`-Syntax verwendet werden kann, um übergeordnete Verzeichnisse zu bezeichnen:

```
var file1:File = File.documentsDirectory;
file1 = file1.resolvePath("AIR Test");
var file2:File = File.documentsDirectory;
file2 = file2.resolvePath("AIR Test/bob/test.txt");
var file3:File = File.documentsDirectory;
file3 = file3.resolvePath("AIR Test/susan/test.txt");

trace(file2.getRelativePath(file1, true)); // ../..
trace(file3.getRelativePath(file2, true)); // ../../bob/test.txt
```

Abrufen der vorschriftsmäßigen Versionen von Dateinamen

Adobe AIR 1.0 und höher

Unter Windows und Mac OS muss die Groß- und Kleinschreibung für Datei- und Pfadnamen nicht beachtet werden. Im Folgenden zeigen zwei File-Objekte auf dieselbe Datei:

```
 File.documentsDirectory.resolvePath("test.txt");
File.documentsDirectory.resolvePath("TeSt.TxT");
```

Allerdings wird in Dokument- und Verzeichnisnamen durchaus Groß- und Kleinschreibung verwendet. Im folgenden Code wird beispielsweise davon ausgegangen, dass sich im Verzeichnis „documents“ ein Ordner mit dem Namen „AIR Test“ befindet:

```
 var file:File = File.documentsDirectory.resolvePath("AIR test");
trace(file.nativePath); // ... AIR test
file.canonicalize();
trace(file.nativePath); // ... AIR Test
```

Die Methode `canonicalize()` konvertiert das Objekt `nativePath`, um die exakte Groß-/Kleinschreibung für den Datei- oder Verzeichnisnamen zu verwenden. Bei Systemen, die zwischen Groß- und Kleinschreibung unterscheiden (wie Linux), korrigiert die `canonicalize()`-Methode für den Fall, dass sich mehrere Dateien nur in der Groß- und Kleinschreibung unterscheiden, den Pfad, um ihn an die erste gefundene Datei anzupassen (die Reihenfolge wird dabei vom System bestimmt).

Unter Windows können Sie mithilfe der Methode `canonicalize()` auch kurze Dateinamen („8.3-Namen“) in lange Dateinamen konvertieren:

```
var path:File = new File();
path.nativePath = "C:\\AIR~1";
path.canonicalize();
trace(path.nativePath); // C:\AIR Test
```

Arbeiten mit Paketen und symbolischen Links

Adobe AIR 1.0 und höher

Verschiedene Betriebssysteme unterstützen Paketdateien und Dateien für symbolische Links:

Pakete – Unter Mac OS können Verzeichnisse als Pakete gekennzeichnet werden und erscheinen dann im Mac OS-Finder als einzelne Dateien, nicht als Verzeichnisse.

Symbolische Vrknüpfungen – Mac OS, Linux und Windows Vista unterstützen symbolische Verknüpfungen. Symbolische Links ermöglichen, dass eine Datei auf eine andere Datei oder ein Verzeichnis auf der Festplatte zeigt. Obwohl sie sich ähneln, sind symbolische Links nicht dasselbe wie Aliasnamen. Ein Alias wird immer als Datei (nicht als Verzeichnis) betrachtet, und das Lesen und Schreiben von oder in einen Alias oder in eine Verknüpfung betrifft nie die ursprüngliche Datei oder das ursprüngliche Verzeichnis, auf die bzw. das verwiesen wird. Abgesehen davon verhält sich ein symbolischer Link genau wie die Datei oder das Verzeichnis, auf die bzw. das verwiesen wird. Der symbolische Link kann als Datei oder als Verzeichnis betrachtet werden. Das Lesen von einem oder Schreiben in einen symbolischen Link betrifft die Datei oder das Verzeichnis, auf die bzw. das verwiesen wird, nicht den symbolischen Link selbst. Unter Windows wird zusätzlich die `isSymbolicLink`-Eigenschaft für ein File-Objekt, das auf einen Verknüpfungspunkt (im NTFS-Dateisystem verwendet) verweist, mit dem Wert `true` belegt.

Die File-Klasse besitzt die Eigenschaften `isPackage` und `isSymbolicLink`, um zu überprüfen, ob ein File-Objekt auf ein Paket oder einen symbolischen Link verweist.

Der folgende Code durchläuft das Desktopverzeichnis des Benutzers und listet alle Unterverzeichnisse auf, die *keine* Pakete sind:

```
var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (desktopNodes[i].isDirectory && !desktopNodes[i].isPackage)
    {
        trace(desktopNodes[i].name);
    }
}
```

Der folgende Code durchläuft das Desktopverzeichnis des Benutzers und listet alle Dateien und Verzeichnisse auf, die *keine* symbolische Links sind:

```
var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (!desktopNodes[i].isSymbolicLink)
    {
        trace(desktopNodes[i].name);
    }
}
```

Die Methode `canonicalize()` ändert den Pfad eines symbolischen Links so, dass er auf die Datei oder das Verzeichnis zeigt, auf die bzw. das der Link verweist. Der folgende Code durchläuft das Desktopverzeichnis des Benutzers und erstellt einen Bericht über alle Pfade, auf die von Dateien verwiesen wird, die symbolische Links sind:

```
var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (desktopNodes[i].isSymbolicLink)
    {
        var linkNode:File = desktopNodes[i] as File;
        linkNode.canonicalize();
        trace(linkNode.nativePath);
    }
}
```

Ermitteln des auf einem Datenträger verfügbaren Speicherplatzes

Adobe AIR 1.0 und höher

Die Eigenschaft `spaceAvailable` eines File-Objekts bezeichnet den Speicherplatz in Byte, der an diesem Dateispeicherort zur Verfügung steht. Im folgenden Beispielcode wird der im Anwendungsspeicherverzeichnis verfügbare Speicherplatz überprüft:

```
trace(File.applicationStorageDirectory.spaceAvailable);
```

Wenn das File-Objekt auf ein Verzeichnis verweist, gibt die `spaceAvailable`-Eigenschaft an, wie viel Speicherplatz die Dateien in diesem Verzeichnis einnehmen können. Wenn das File-Objekt auf eine Datei verweist, gibt die Eigenschaft `spaceAvailable` an, wie groß die Datei werden kann. Wenn der Dateispeicherort nicht vorhanden ist, wird die Eigenschaft `spaceAvailable` auf 0 eingestellt. Wenn das File-Objekt auf einen symbolischen Link verweist, entspricht der Wert, auf den die Eigenschaft `spaceAvailable`gesetzt wird, dem verfügbaren Speicherplatz an dem Speicherort, auf den der symbolische Link verweist.

Normalerweise ist der für ein Verzeichnis oder eine Datei verfügbare Speicherplatz identisch mit dem Speicherplatz, der auf dem Datenträger mit der Datei bzw. dem Verzeichnis verfügbar ist. Der verfügbare Speicherplatz kann jedoch auch Speicherkontingente und verzeichnisspezifische Limits berücksichtigen.

Wenn Sie einem Datenträger eine Datei oder ein Verzeichnis hinzufügen, wird dabei im Allgemeinen mehr Speicherplatz benötigt, als die Datei bzw. der Inhalt des Verzeichnisses tatsächlich belegen. Das Betriebssystem kann zum Beispiel zusätzlichen Speicherplatz zum Speichern von Indexinformationen benötigen. Auch Festplattensektoren benötigen unter Umständen zusätzlichen Speicher. Der verfügbare Speicherplatz ändert sich zudem dynamisch. Sie können also nicht davon ausgehen, dass Sie den gesamten gemeldeten Speicherplatz für das Speichern von Dateien verwenden können. Informationen dazu, wie Sie ins Dateisystem schreiben, finden Sie unter „Lese- und Schreibvorgänge in Dateien“ auf Seite 735.

Die `StorageVolumeInfo.getStorageVolumes()`-Methode liefert ausführlichere Informationen über gemountete Speichermedien (siehe „Arbeiten mit Speichermedien“ auf Seite 733).

Öffnen von Dateien mit der Standardsystemanwendung

Adobe AIR 2 und höher

In AIR 2 können Sie eine Datei mit der Anwendung öffnen, die im Betriebssystem zum Öffnen der jeweiligen Datei registriert ist. Beispielsweise kann eine AIR-Anwendung eine DOC-Datei mit der dafür registrierten Anwendung öffnen. Verwenden Sie die `openWithDefaultApplication()`-Methode eines File-Objekts zum Öffnen der Datei. Mit dem folgenden Code wird beispielsweise eine Datei namens „test.doc“ auf dem Desktop des Anwenders mit der Standardanwendung für DOC-Dateien geöffnet:

```
var file:File = File.deskopDirectory;
file = file.resolvePath("test.doc");
file.openWithDefaultApplication();
```

Hinweis: *Unter Linux bestimmt nicht die Dateierweiterung, sondern der MIME-Typ die Standardanwendung für eine Datei.*

Mit dem folgenden Code können Sie zu einer MP3-Datei navigieren und sie in der Standardanwendung zum Abspielen von MP3-Dateien öffnen.

```
var file:File = File.documentsDirectory;
var mp3Filter:FileFilter = new FileFilter("MP3 Files", "*.mp3");
file.browseForOpen("Open", [mp3Filter]);
file.addEventListener(Event.SELECT, fileSelected);

function fileSelected(e:Event):void
{
    file.openWithDefaultApplication();
}
```

Die `openWithDefaultApplication()`-Methode kann nicht mit Dateien verwendet werden, die sich im Anwendungsverzeichnis befinden.

AIR verhindert das Öffnen bestimmter Dateien mit der `openWithDefaultApplication()`-Methode. Unter Windows verhindert AIR das Öffnen von Dateien mit bestimmten Dateitypen, wie beispielsweise EXE oder BAT. Unter Mac OS und Linux verhindert AIR, dass Sie Dateien öffnen, die in bestimmten Anwendungen gestartet werden. (Dazu zählen Terminal und AppletLauncher unter Mac OS sowie csh, bash und ruby unter Linux.) Wenn Sie versuchen, eine dieser Dateien mit der `openWithDefaultApplication()`-Methode zu öffnen, wird ein Ausnahmefehler ausgegeben. Eine vollständige Liste der Dateitypen, die nicht geöffnet werden können, finden Sie im Referenzhandbuch im Eintrag für die `File.openWithDefaultApplication()`-Methode.

Hinweis: *Diese Einschränkung gilt nicht für AIR-Anwendungen, die über ein natives Installationsprogramm installiert wurden (erweiterte Desktop-Anwendung).*

Abrufen von Dateisysteminformationen

Adobe AIR 1.0 und höher

Die File-Klasse umfasst die folgenden statischen Eigenschaften, die nützliche Informationen über das Dateisystem bereitstellen:

Eigenschaft	Beschreibung
`File.lineEnding`	Die vom Host-Betriebssystem verwendete Zeilenende-Zeichenfolge. Unter Mac OS und Linux ist dies das Zeichen für den Zeilenvorschub. Unter Windows ist dies das Wagenrücklaufzeichen, gefolgt vom Zeilenvorschubzeichen.
`File.separator`	Das vom Betriebssystem verwendete Trennzeichen für Pfadkomponenten. Unter Mac OS und Linux ist dies der Schrägstrich (/). Unter Windows ist dies der umgekehrte Schrägstrich (\).
`File.systemCharset`	Die vom Host-Betriebssystem für Dateien verwendete Standardkodierung. Diese Angabe bezieht sich auf den vom Betriebssystem für eine bestimmte Sprache verwendeten Zeichensatz.

Die `Capabilities`-Klasse bietet ebenfalls nützliche Systeminformationen, die für die Arbeit mit Dateien praktisch sind:

Eigenschaft	Beschreibung
Capabilities.hasIME	Gibt an, ob der Player auf einem System ausgeführt wird, auf dem ein Eingabemethoden-Editor (IME) installiert (`true`) oder nicht installiert (`false`) ist.
Capabilities.language	Gibt den Sprachcode des Systems an, auf dem der Player ausgeführt wird.
Capabilities.os	Gibt das aktuelle Betriebssystem an.

Hinweis: *Bei der Verwendung von `Capabilities.os` zur Bestimmung von Systemmerkmalen ist Vorsicht geboten. Falls es eine spezifischere Eigenschaft zur Bestimmung eines Systemmerkmals vorhanden ist, verwenden Sie diese. Andernfalls besteht die Gefahr, dass Sie Code schreiben, der nicht auf allen Plattformen korrekt funktioniert. Betrachten Sie den folgenden Beispielcode:*

```
var separator:String;
if (Capablities.os.indexOf("Mac") > -1)
{
    separator = "/";
}
else
{
    separator = "\\";
}
```

Dieser Code führt unter Linux zu Problemen. Es ist besser, einfach die `File.separator`-Eigenschaft zu verwenden.

Arbeiten mit Verzeichnissen

Adobe AIR 1.0 und höher

Die Laufzeitumgebung stellt Ihnen Möglichkeiten bereit, um mit Verzeichnissen auf dem lokalen Dateisystem zu arbeiten.

Erläuterungen zum Erstellen von File-Objekten, die auf Verzeichnisse verweisen, finden Sie unter „Verweisen vom File-Objekt auf ein Verzeichnis“ auf Seite 715.

Erstellen von Verzeichnissen

Adobe AIR 1.0 und höher

Mithilfe der Methode `File.createDirectory()` können Sie ein Verzeichnis erstellen. Im folgenden Code beispielsweise wird ein Verzeichnis mit dem Namen „AIR Test“ als Unterverzeichnis des Stammverzeichnisses eines Benutzers erstellt:

```
var dir:File = File.userDirectory.resolvePath("AIR Test");
dir.createDirectory();
```

Ist das Verzeichnis bereits vorhanden, hat die Methode `createDirectory()` keine Auswirkung.

In einigen Modi erstellt ein FileStream-Objekt beim Öffnen von Dateien ein Verzeichnis. Fehlende Verzeichnisse werden erstellt, wenn Sie eine FileStream-Instanz instantiieren und dabei den Parameter `fileMode` des `FileStream()`-Konstruktors auf `FileMode.APPEND` oder `FileMode.WRITE` setzen. Weitere Informationen finden Sie unter „Workflow zu Lese- und Schreibvorgängen in Dateien“ auf Seite 735.

Erstellen eines temporären Verzeichnisses

Adobe AIR 1.0 und höher

Die File-Klasse beinhaltet die Methode `createTempDirectory()`, die im Systemordner für temporäre Dateien ein Verzeichnis anlegt. Hier ein Beispiel:

```
var temp:File = File.createTempDirectory();
```

Die Methode `createTempDirectory()` erstellt automatisch ein eindeutiges temporäres Verzeichnis (sodass Sie nicht selbst einen eindeutigen Pfad ermitteln müssen).

In einem temporären Verzeichnis können Sie Dateien vorübergehend speichern, die nur während der aktuellen Anwendungssitzung benötigt werden. Beachten Sie, dass es eine neue `createTempFile()`-Methode gibt, um im Systemordner für temporäre Dateien neue, eindeutige Dateien zu erstellen.

Da das temporäre Verzeichnis *nicht* auf allen Geräten automatisch gelöscht wird, sollten Sie es evtl. löschen, bevor die Anwendung geschlossen wird.

Erstellen von Verzeichnislisten

Adobe AIR 1.0 und höher

Mithilfe der Methoden `getDirectoryListing()` und `getDirectoryListingAsync()` eines File-Objekts können Sie einen Array mit File-Objekten abrufen, die auf Dateien und Unterverzeichnisse in einem bestimmten Verzeichnis zeigen.

Im folgenden Beispielcode wird der Inhalt des Benutzerdokumentverzeichnisses aufgelistet (ohne Untersuchung der Unterverzeichnisse):

```
var directory:File = File.documentsDirectory;
var contents:Array = directory.getDirectoryListing();
for (var i:uint = 0; i < contents.length; i++)
{
    trace(contents[i].name, contents[i].size);
}
```

Wenn Sie die asynchrone Version der Methode verwenden, besitzt das Ereignisobjekt `directoryListing` eine `files`-Eigenschaft, welche ein Array an File-Objekten, die zum betreffenden Verzeichnis gehören, darstellt:

```
var directory:File = File.documentsDirectory;
directory.getDirectoryListingAsync();
directory.addEventListener(FileListEvent.DIRECTORY_LISTING, dirListHandler);

function dirListHandler(event:FileListEvent):void
{
    var contents:Array = event.files;
    for (var i:uint = 0; i < contents.length; i++)
    {
        trace(contents[i].name, contents[i].size);
    }
}
```

Kopieren und Verschieben von Verzeichnissen

Adobe AIR 1.0 und höher

Sie können ein Verzeichnis mit denselben Methode kopieren oder verschieben, die Sie auch zum Kopieren und Verschieben von Dateien verwenden. Im folgenden Beispielcode wird ein Verzeichnis synchron kopiert:

```
var sourceDir:File = File.documentsDirectory.resolvePath("AIR Test");
var resultDir:File = File.documentsDirectory.resolvePath("AIR Test Copy");
sourceDir.copyTo(resultDir);
```

Wenn Sie für den Parameter `overwrite` der Methode `copyTo()` den Wert „true“ angeben, werden alle Dateien und Ordner in einem vorhandenen Zielverzeichnis durch die Dateien und Ordner im Quellverzeichnis ersetzt (auch wenn die Zieldatei im Quellverzeichnis nicht existiert).

Das Verzeichnis, das Sie im Parameter `newLocation` der Methode `copyTo()` angeben, bestimmt den Pfad zum Ergebnisverzeichnis, aber *nicht* das *übergeordnete* Verzeichnis, in dem das Ergebnisverzeichnis enthalten sein wird.

Weitere Informationen finden Sie unter „Kopieren und Verschieben von Dateien“ auf Seite 731.

Löschen des Verzeichnisinhalts

Adobe AIR 1.0 und höher

Die File-Klasse umfasst eine `deleteDirectory()`- und eine `deleteDirectoryAsync()`-Methode. Beide Methoden löschen Verzeichnisse, wobei die erste synchron arbeitet und die zweite asynchron (siehe auch „Grundlegende Dateioperationen“ auf Seite 711). Auch besitzen beide den booleschen Parameter `deleteDirectoryContents`. Ist dieser auf `true` gesetzt, löscht der Methodenaufruf nicht-leere Verzeichnisse. Ist er auf `false` gesetzt (der Standardwert), werden nur leere Verzeichnisse gelöscht.

Im folgenden Code beispielsweise wird das Unterverzeichnis „AIR Test“ synchron aus dem Dokumentenverzeichnis des Benutzers gelöscht:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.deleteDirectory(true);
```

Im folgenden Code wird das Unterverzeichnis „AIR Test“ asynchron aus dem Dokumentenverzeichnis des Benutzers gelöscht:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.addEventListener(Event.COMPLETE, completeHandler)
directory.deleteDirectoryAsync(true);

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Die Klasse umfasst zudem die Methoden `moveToTrash()` und `moveToTrashAsync()`, mit denen Sie ein Verzeichnis in den Papierkorb des Systems verschieben können. Weitere Informationen finden Sie unter „Verschieben einer Datei in den Papierkorb“ auf Seite 733.

Arbeiten mit Dateien

Adobe AIR 1.0 und höher

Mit der AIR-Datei-API können Sie grundlegende Funktionen zur Interaktion mit Dateien in Ihre Anwendungen einbinden. So können Sie etwa Dateien lesen und in Dateien schreiben, Dateien kopieren, löschen usw. Da Ihre Anwendungen auf das lokale Dateisystem zugreifen können, sollten Sie den Abschnitt „AIR-Sicherheit“ auf Seite 1153 lesen.

Hinweis: *Sie haben die Möglichkeit, einen Dateityp mit einer AIR-Anwendung zu verknüpfen (sodass die Anwendung bei Doppelklick auf eine Datei dieses Typs geöffnet wird). Weitere Informationen finden Sie unter „Verwalten von Dateiverknüpfungen“ auf Seite 945.*

Abrufen von Datei-Informationen

Adobe AIR 1.0 und höher

Die File-Klasse umfasst die folgenden Eigenschaften, die Informationen über die Datei oder das Verzeichnis bereitstellen, auf die ein File-Objekt zeigt:

Eigenschaft	Beschreibung
creationDate	Das Erstellungsdatum der Datei auf der lokalen Festplatte.
creator	Veraltet. – Verwenden Sie die Eigenschaft `extension`. (Diese Eigenschaft nennt den Macintosh-Erstellertyp der Datei und wird nur mit Mac OS-Versionen, die älter als Mac OS X sind, verwendet.)
downloaded	(AIR 2 und höher) Gibt an, ob die referenzierte Datei bzw. das referenzierte Verzeichnis (aus dem Internet) heruntergeladen wurde oder nicht. Diese Eigenschaft ist nur unter Betriebssystemen, in denen Dateien als heruntergeladen gekennzeichnet werden können, von Bedeutung: • Windows XP Service Pack 2 und höher und Windows Vista • Mac OS 10.5 und höher
exists	Diese Eigenschaft gibt an, ob die Datei bzw. das Verzeichnis, auf die bzw. das verwiesen wird, vorhanden ist.

Eigenschaft	Beschreibung
extension	Die Dateierweiterung, d. h. der Teil des Namens, der hinter dem letzten Punkt (.) steht. Der Punkt selbst gehört nicht zur Erweiterung. Wenn der Dateiname keinen Punkt enthält, ist die Erweiterung `null`.
icon	Ein Icon-Objekt, das die für diese Datei definierten Symbole enthält.
isDirectory	Gibt an, ob das File-Objekt auf ein Verzeichnis verweist.
modificationDate	Das Datum, an dem die Datei bzw. das Verzeichnis auf der lokalen Festplatte zuletzt geändert wurde.
name	Der Name der Datei bzw. des Verzeichnisses auf der lokalen Festplatte (einschließlich Dateierweiterung, falls vorhanden).
nativePath	Der vollständige Pfad in der Schreibweise des Host-Betriebssystems (siehe „Pfade von File-Objekten“ auf Seite 712).
parent	Der Ordner, der den vom File-Objekt repräsentierten Ordner bzw. die Datei enthält. Die Eigenschaft hat den Wert `null`, wenn das File-Objekt auf eine Datei oder ein Verzeichnis im Stammverzeichnis des Dateisystems verweist.
size	Die Größe der Datei auf der lokalen Festplatte in Byte.
type	Veraltet. – Verwenden Sie die Eigenschaft `extension`. (Auf Macintosh-Rechnern ist der Wert dieser Eigenschaft ein Dateityp aus vier Zeichen, wird aber nur mit Mac OS-Versionen, die älter als Mac OS X sind, verwendet.)
url	Die URL der Datei bzw. des Verzeichnisses. (siehe „Pfade von File-Objekten“ auf Seite 712).

Einzelheiten zu diesen Eigenschaften finden Sie im Eintrag zur File-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Kopieren und Verschieben von Dateien

Adobe AIR 1.0 und höher

Die File-Klasse umfasst zwei Methoden für das Kopieren von Dateien und Verzeichnissen: `copyTo()` und `copyToAsync()`. Die File-Klasse umfasst zwei Methoden für das Verschieben von Dateien und Verzeichnissen: `moveTo()` und `moveToAsync()`. Die Methoden `copyTo()` und `moveTo()` arbeiten synchron, während die Methoden `copyToAsync()` und `moveToAsync()` asynchron arbeiten (siehe „Grundlegende Dateioperationen“ auf Seite 711).

Um eine Datei zu kopieren oder zu verschieben, richten Sie zwei File-Objekte ein. Das eine zeigt auf die zu kopierende bzw. zu verschiebende Datei und ruft die Methode auf. Das andere zeigt auf den Zielpfad (auch „Ergebnispfad“ genannt).

Im Folgenden wird die Datei „test.txt“ vom Unterverzeichnis „AIR Test“ des Dokumentenverzeichnisses des Benutzers in eine Datei mit dem Namen „copy.txt“ desselben Verzeichnisses kopiert:

```
var original:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var newFile:File = File.resolvePath("AIR Test/copy.txt");
original.copyTo(newFile, true);
```

In diesem Beispiel wird der Wert des Parameters `overwrite` der Methode `copyTo()` (der zweite Parameter) auf `true` gesetzt. Indem Sie `overwrite` auf `true` setzen, wird eine eventuell vorhandene Zieldatei überschrieben. Dieser Parameter ist optional. Wenn Sie ihn auf `false` setzen (den Standardwert), löst die Operation ein IOErrorEvent-Ereignis aus, wenn die Zieldatei bereits existiert (und die Datei wird nicht kopiert).

Die „Async“-Versionen für das Kopieren und Verschieben arbeiten asynchron. Verwenden Sie die `addEventListener()`-Methode wie folgt, um die Fertigstellung der Aufgabe oder Fehlerbedingung zu überwachen:

```
var original = File.documentsDirectory;
original = original.resolvePath("AIR Test/test.txt");

var destination:File = File.documentsDirectory;
destination =  destination.resolvePath("AIR Test 2/copy.txt");

original.addEventListener(Event.COMPLETE, fileMoveCompleteHandler);
original.addEventListener(IOErrorEvent.IO_ERROR, fileMoveIOErrorEventHandler);
original.moveToAsync(destination);

function fileMoveCompleteHandler(event:Event):void {
    trace(event.target); // [object File]
}
function fileMoveIOErrorEventHandler(event:IOErrorEvent):void {
    trace("I/O Error.");
}
```

Die File-Klasse umfasst zudem die Methoden `File.moveToTrash()` und `File.moveToTrashAsync()`, die eine Datei oder ein Verzeichnis in den Papierkorb des Systems verschieben.

Löschen von Dateien

Adobe AIR 1.0 und höher

Die File-Klasse umfasst eine `deleteFile()`- und eine `deleteFileAsync()`-Methode. Beide Methoden löschen Dateien, wobei die erste synchron arbeitet und die zweite asynchron (siehe auch „Grundlegende Dateioperationen“ auf Seite 711).

Im folgenden Code beispielsweise wird die Datei „test.txt“ synchron aus dem Dokumentenverzeichnis des Benutzers gelöscht:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.deleteFile();
```

Im folgenden Code wird das Unterverzeichnis „AIR Test“ asynchron aus dem Dokumentenverzeichnis des Benutzers gelöscht:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, completeHandler)
file.deleteFileAsync();

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Die Klasse umfasst zudem die Methoden `moveToTrash()` und `moveToTrashAsync()`, mit denen Sie eine Datei oder ein Verzeichnis in den Papierkorb des Systems verschieben können. Weitere Informationen finden Sie unter „Verschieben einer Datei in den Papierkorb“ auf Seite 733.

Verschieben einer Datei in den Papierkorb

Adobe AIR 1.0 und höher

Die File-Klasse umfasst eine `moveToTrash()`- und eine `moveToTrashAsync()`-Methode. Beide Methoden senden eine Datei oder ein Verzeichnis an den Papierkorb des Systems, wobei die erste synchron arbeitet und die zweite asynchron (siehe auch „Grundlegende Dateioperationen“ auf Seite 711).

Im folgenden Code beispielsweise wird die Datei „test.txt“ synchron vom Dokumentenverzeichnis des Benutzers in den Systempapierkorb verschoben:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.moveToTrash();
```

Hinweis: *Bei Betriebssystemen, die keinen wiederherstellbaren Papierkorb haben, werden die Dateien sofort gelöscht.*

Erstellen einer temporären Datei

Adobe AIR 1.0 und höher

Die File-Klasse beinhaltet die Methode `createTempFile()`, die im Systemordner für temporäre Dateien eine Datei anlegt. Hier ein Beispiel:

```
var temp:File = File.createTempFile();
```

Die Methode `createTempFile()` erstellt automatisch eine eindeutige temporäre Datei (sodass Sie nicht selbst einen eindeutigen Pfad ermitteln müssen).

In einer temporären Datei können Sie vorübergehend Informationen speichern, die nur während der aktuellen Anwendungssitzung benötigt werden. Beachten Sie, dass es eine neue `createTempDirectory()`-Methode gibt, um im Systemordner für temporäre Dateien neue, eindeutige Verzeichnisse zu erstellen.

Da die temporäre Datei *nicht* auf allen Geräten automatisch gelöscht wird, sollten Sie sie evtl. löschen, bevor die Anwendung geschlossen wird.

Arbeiten mit Speichermedien

Adobe AIR 2 und höher

In AIR 2 kann das Mounten und Unmounten von Massenspeichermedien erkannt werden. Die StorageVolumeInfo-Klasse definiert ein `storageVolumeInfo`-Singleton-Objekt. Das `StorageVolumeInfo.storageVolumeInfo`-Objekt setzt ein `storageVolumeMount`-Ereignis ab, wenn ein Speichermedium gemountet wird. Beim Unmounten eines Speichermediums setzt es ein `storageVolumeUnmount`-Ereignis ab. Die StorageVolumeChangeEvent-Klasse definiert diese Ereignisse.

Hinweis: *In modernen Linux-Distributionen setzt das StorageVolumeInfo-Objekt nur `storageVolumeMount`- und `storageVolumeUnmount`-Ereignisse für physische Geräte und Netzwerkgeräte ab, die an bestimmten Orten bereitgestellt werden.*

Die `storageVolume`-Eigenschaft der StorageVolumeChangeEvent-Klasse ist ein StorageVolume-Objekt. Die StorageVolume-Klasse definiert grundlegende Eigenschaften des Speichermediums:

- `drive` – Der Buchstabe des Speichermediums unter Windows. (`null` auf anderen Betriebssystemen)
- `fileSystemType` – Das Dateisystem des Speichermediums (wie FAT, NTFS, HFS oder UFS).
- `isRemoveable` – Gibt an, ob es sich um ein Wechselspeichermedium handelt (`true`) oder nicht (`false`).

- `isWritable` – Gibt an, ob ein Speichermedium beschrieben werden kann (`true`) oder nicht (`false`).
- `name` – Der Name des Speichermediums.
- `rootDirectory` – Ein File-Objekt, das dem Stammverzeichnis des Speichermediums entspricht.

Die StorageVolumeChangeEvent-Klasse enthält auch die `rootDirectory`-Eigenschaft. Die `rootDirectory`-Eigenschaft ist ein File-Objekt, das auf das Stammverzeichnis des Speichermediums verweist, für das ein Mounten oder Unmounten durchgeführt wurde.

Die `storageVolume`-Eigenschaft des StorageVolumeChangeEvent-Objekts ist für ein nicht gemountetes Speichermedium nicht definiert (`null`). Sie können jedoch auf die `rootDirectory`-Eigenschaft des Ereignisses zugreifen.

Der folgende Code gibt den Namen und den Dateipfad eines Speichermediums beim Mounten aus:

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_MOUNT, onVolumeMount);
function onVolumeMount(event:StorageVolumeChangeEvent):void
{
    trace(event.storageVolume.name, event.rootDirectory.nativePath);
}
```

Der folgende Code gibt den Dateipfad eines Speichermediums beim Unmounten aus:

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_UNMOUNT, onVolumeUnmount);
function onVolumeUnmount(event:StorageVolumeChangeEvent):void
{
    trace(event.rootDirectory.nativePath);
}
```

Das `StorageVolumeInfo.storageVolumeInfo`-Objekt enthält eine `getStorageVolumes()`-Methode. Diese Methode gibt einen Vektor der StorageVolume-Objekte zurück, die den derzeit gemounteten Speichermedien entsprechen. Mit dem folgenden Code werden die Namen und Stammverzeichnisse aller gemounteten Speichermedien aufgelistet:

```
var volumes:Vector.<StorageVolume> = new Vector.<StorageVolume>;
volumes = StorageVolumeInfo.storageVolumeInfo.getStorageVolumes();
for (var i:int = 0; i < volumes.length; i++)
{
    trace(volumes[i].name, volumes[i].rootDirectory.nativePath);
}
```

Hinweis: *Auf modernen Linux-Distributionen gibt die `getStorageVolumes()`-Methode Objekte zurück, die physischen Geräten und an bestimmten Orten gemounteten Netzwerklaufwerken entsprechen.*

Die `File.getRootDirectories()`-Methode listet die Stammverzeichnisse auf (siehe „Verweisen auf das Stammverzeichnis des Dateisystems“ auf Seite 717). Das StorageVolume-Objekt (von der `StorageVolumeInfo.getStorageVolumes()`-Methode aufgezählt) liefert jedoch mehr Informationen über die Speichermedien.

Mit der `spaceAvailable`-Eigenschaft der `rootDirectory`-Eigenschaft eines StorageVolume-Objekts kann der verfügbare Speicherplatz auf einem Speichermedium festgestellt werden. (Siehe „Ermitteln des auf einem Datenträger verfügbaren Speicherplatzes“ auf Seite 725.)

Verwandte Hilfethemen

StorageVolume

StorageVolumeInfo

Lese- und Schreibvorgänge in Dateien

Adobe AIR 1.0 und höher

Über die FileStream-Klasse können AIR-Anwendungen Daten aus dem Dateisystem lesen und in das Dateisystem schreiben.

Workflow zu Lese- und Schreibvorgängen in Dateien

Adobe AIR 1.0 und höher

Der Workflow zum Lesen aus und Schreiben in Dateien ist wie folgt:

Initialisieren eines File-Objekts, das auf den Pfad zeigt

Das File-Objekt repräsentiert den Pfad der Datei, mit der Sie arbeiten möchten (oder einer Datei, die Sie erst später erstellen).

```
var file:File = File.documentsDirectory;
file = file.resolvePath("AIR Test/testFile.txt");
```

Im folgenden Beispiel werden die `File.documentsDirectory`-Eigenschaft und die `resolvePath()`-Methode eines File-Objekts verwendet, um das File-Objekt zu initialisieren. Allerdings gibt es viele andere Möglichkeiten, um von einem File-Objekt zu einer Datei zu verweisen. Weitere Informationen hierzu finden Sie unter „Verweisen vom File-Objekt zu einer Datei“ auf Seite 719.

Initialisieren eines FileStream-Objekts

Aufrufen der open()- oder openAsync()-Methode des FileStream-Objekts

Welche Methode Sie aufrufen, hängt davon ab, ob Sie die Datei für synchrone oder asynchrone Operationen verwenden wollen. Verwenden Sie das File-Objekt als den `file`-Parameter der open-Methode. Geben Sie für den `fileMode`-Parameter eine Konstante der FileMode-Klasse an, die festlegt, wie Sie die Datei einsetzen.

Der folgende Code etwa initialisiert ein FileStream-Objekt, das zum Erstellen einer Datei und zum Überschreiben etwaiger Daten eingesetzt wird:

```
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);
```

Weitere Informationen finden Sie unter „Initialisieren von FileStream-Objekten (Öffnen und Schießen von Dateien)“ auf Seite 737 und „Modi beim Öffnen von FileStream-Objekten“ auf Seite 736.

Hinzufügen und Einrichten von Ereignis-Listenern für das FileStream-Objekt, wenn Sie die Datei asynchron geöffnet haben (mit der Methode openAsync())

Diese Ereignis-Listener-Methoden reagieren auf Ereignisse, die in verschiedenen Situationen vom FileStream-Objekt ausgelöst werden, beispielsweise wenn Daten aus der Datei gelesen werden, wenn E/A-Fehler auftreten oder wenn die Datenmenge vollständig geschrieben wurde.

Weitere Informationen finden Sie unter „Asynchrone Programmierung und Ereignisse“ auf Seite 741.

Gegebenenfalls Einbinden von Code zum Lesen und Schreiben von Daten

Viele Methoden der FileStream-Klasse haben mit Lese- und Schreibvorgängen zu tun. (Ihr Name beginnt immer mit „read“ oder „write“.) Welche Methode Sie zum Lesen oder Schreiben von Daten verwenden, hängt vom Format der Daten in der Zieldatei ab.

Wenn es sich bei den Daten in der Zieldatei beispielsweise um UTF-kodierten Text handelt, können Sie die Methoden `readUTFBytes()` und `writeUTFBytes()` verwenden. Wenn Sie die Daten als Byte-Arrays weiterverarbeiten wollen, können Sie die Methoden `readByte()`, `readBytes()`, `writeByte()` und `writeBytes()` verwenden. Ausführliche Informationen finden Sie unter „Datenformate und die geeigneten read- und write-Methoden“ auf Seite 742.

Wenn Sie die Datei asynchron geöffnet haben, stellen Sie vor dem Aufruf eine read-Methode sicher, dass genügend Daten zur Verfügung stehen. Ausführliche Informationen finden Sie unter „Der Lesepuffer und die bytesAvailable-Eigenschaft“ auf Seite 739.

Bevor Sie in eine Datei schreiben, können Sie anhand der spaceAvailable-Eigenschaft des File-Objekts überprüfen, wie viel Plattenspeicher zur Verfügung steht. Weitere Informationen hierzu finden Sie unter „Ermitteln des auf einem Datenträger verfügbaren Speicherplatzes“ auf Seite 725.

Aufrufen der close()-Methode des FileStream-Objekts, wenn die Arbeit mit der Datei abgeschlossen ist.

Durch Aufruf der close()-Methode steht die Datei anderen Anwendungen zur Verfügung.

Ausführliche Informationen hierzu finden Sie unter „Initialisieren von FileStream-Objekten (Öffnen und Schießen von Dateien)“ auf Seite 737.

Beispielanwendungen, in denen für das Lesen und Schreiben von Dateien die FileStream-Klasse verwendet wird, finden Sie in den folgenden Artikeln des Adobe AIR Developer Centers:

- Erstellen eines Textdatei-Editors
- Erstellen eines Textdatei-Editors
- Lesen und Schreiben von einer XML-Einstellungsdatei

Arbeiten mit FileStream-Objekten

Adobe AIR 1.0 und höher

Die FileStream-Klasse definiert Methoden für das Öffnen, Lesen und Schreiben von Dateien.

Modi beim Öffnen von FileStream-Objekten

Adobe AIR 1.0 und höher

Die Methoden `open()` und `openAsync()` eines FileStream-Objekts besitzen beide den Parameter `fileMode`. Dieser definiert mehrere Eigenschaften für einen Dateistream, u. a. die folgenden:

- Die Möglichkeit aus einer Datei zu lesen
- Die Möglichkeit in eine Datei zu schreiben
- Werden bei Schreibvorgängen die Daten immer an das Ende der Datei geschrieben?
- Was geschieht, wenn eine Datei und ihre übergeordneten Verzeichnisse nicht existieren?

Es folgt eine Auflistung der verschiedenen Dateimodi (die Sie als `fileMode`-Parameter der Methoden `open()` und `openAsync()` angeben können):

Modus	Beschreibung
FileMode.READ	Gibt an, dass eine Datei nur zum Lesen geöffnet wird.
FileMode.WRITE	Gibt an, dass eine Datei mit Schreibrechten geöffnet wird. Sollte die Datei nicht existieren, wird sie beim Öffnen des FileStream-Objekts erstellt. Wenn die Datei existiert, werden vorhandene Daten gelöscht.
FileMode.APPEND	Gibt an, dass eine Datei im Ergänzungsmodus geöffnet wird. Wenn die Datei nicht existiert, wird sie angelegt. Wenn die Datei existiert, werden vorhandene Daten nicht überschriebenen, und alle neuen Daten werden an das Ende der Datei angehängt.
FileMode.UPDATE	Gibt an, dass eine Datei mit Lese- und Schreibrechten geöffnet wird. Sollte die Datei nicht existieren, wird sie erstellt. Dieser Modus eignet sich für zufällige Lese-/Schreibzugriffe auf eine Datei. Sie können von jeder Position in der Datei aus Daten lesen. Beim Schreiben in die Datei werden die vorhandenen Byte nur durch die geschriebenen ersetzt (alle anderen bleiben unverändert).

Initialisieren von FileStream-Objekten (Öffnen und Schießen von Dateien)

Adobe AIR 1.0 und höher

Indem Sie ein FileStream-Objekt öffnen, geben Sie ihm die Möglichkeit, Daten aus einer Datei zu lesen und in eine Datei zu schreiben. Sie öffnen ein FileStream-Objekt, indem Sie an seine `open()` oder `openAsync()`-Methode ein File-Objekt übergeben:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
```

Der `fileMode`-Parameter (der zweite Parameter der Methode `open()` bzw. `openAsync()`) legt den Modus fest, in dem eine Datei zu Öffnen ist: Lesen (read), Schreiben (write), Ergänzen (append) oder Aktualisieren (update). Ausführliche Erläuterungen erhalten Sie im obigen Abschnitt, „Modi beim Öffnen von FileStream-Objekten“ auf Seite 736.

Wenn Sie die `openAsync()`-Methode verwenden, um die Datei für asynchrone Dateioperationen zu öffnen, richten Sie Ereignis-Listener ein, die die asynchronen Ereignisse verarbeiten:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completeHandler);
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.addEventListener(IOErrorEvent.IO_Error, errorHandler);
myFileStream.openAsync(myFile, FileMode.READ);

function completeHandler(event:Event):void {
    // ...
}

function progressHandler(event:ProgressEvent):void {
    // ...
}

function errorHandler(event:IOErrorEvent):void {
    // ...
}
```

Die Datei wird für synchrone oder asynchrone Operationen geöffnet, abhängig davon, ob Sie die `open()`- oder `openAsync()`-Methode verwenden. Ausführliche Informationen finden Sie unter „Grundlegende Dateioperationen“ auf Seite 711.

Wenn Sie in der open-Methode des FileStream-Objekts den Parameter `fileMode` auf `FileMode.READ` oder `FileMode.UPDATE` setzen, werden die Daten in den Lesepuffer gelesen, sobald Sie das FileStream-Objekt öffnen. Ausführliche Informationen finden Sie unter „Der Lesepuffer und die bytesAvailable-Eigenschaft“ auf Seite 739.

Sie können die `close()`-Methode eines FileStream-Objekts aufrufen, um die verknüpfte Datei zu schließen und anderen Anwendungen zur Verfügung zu stellen.

Die position-Eigenschaft eines FileStream-Objekts

Adobe AIR 1.0 und höher

Die `position`-Eigenschaft eines FileStream-Objekts legt fest, wo Daten mit der nächsten read- oder write-Methode gelesen werden.

Setzen Sie vor einer Lese- oder Schreiboperation die Eigenschaft `position` auf eine beliebige, gültige Position in der Datei.

Der folgende Code beispielsweise schreibt an der Position 8 in der Datei den String `"hello"` (in UTF-Kodierung):

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.UPDATE);
myFileStream.position = 8;
myFileStream.writeUTFBytes("hello");
```

Wenn Sie ein FileStream-Objekt das erste Mal öffnen, wird die Eigenschaft `position` auf 0 gesetzt.

Vor einer read-Operation muss der Wert von `position` mindestens 0 sein und weniger als die Anzahl von Byte in der Datei (d. h. vorhandene Positionen in der Datei).

Der Wert der Eigenschaft `position` wird nur in den folgenden Situationen verändert:

- Wenn Sie die Eigenschaft `position` ausdrücklich festlegen.
- Wenn Sie eine read-Methode aufrufen.
- Wenn Sie eine write-Methode aufrufen.

Wenn Sie eine read- oder write-Methode eines FileStream-Objekts aufrufen, wird die Eigenschaft `position` sofort um die Anzahl von Byte hochgezählt, die Sie lesen oder schreiben. Abhängig davon, welche read-Methode Sie verwenden, wird die Eigenschaft `position` entweder um die Anzahl von Byte hochgezählt, die Sie lesen oder schreiben, oder um die Anzahl verfügbarer Byte. Wenn Sie anschließend eine read- oder write-Methode aufrufen, wird ab der neuen Position gelesen oder geschrieben.

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.UPDATE);
myFileStream.position = 4000;
trace(myFileStream.position); // 4000
myFileStream.writeBytes(myByteArray, 0, 200);
trace(myFileStream.position); // 4200
```

Es gibt jedoch eine Ausnahme: bei einem im Ergänzungsmodus geöffneten FileStream-Objekt wird die Eigenschaft `position` nach einem Aufruf der write-Methode nicht geändert. (Im Ergänzungsmodus werden die Daten immer ans Dateiende angehängt, unabhängig vom Wert der `position`-Eigenschaft.)

Bei einer Datei, die für asynchrone Operationen geöffnet wurde, wird die write-Operation erst nach der Ausführung der nächsten Codezeile abgeschlossen. Sie können jedoch mehrere asynchrone Methoden nacheinander aufrufen und die Laufzeitumgebung wird sie in der angeführten Reihenfolge ausführen:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.openAsync(myFile, FileMode.WRITE);
myFileStream.writeUTFBytes("hello");
myFileStream.writeUTFBytes("world");
myFileStream.addEventListener(Event.CLOSE, closeHandler);
myFileStream.close();
trace("started.");

closeHandler(event:Event):void
{
    trace("finished.");
}
```

Die Trace-Ausgabe für diesen Code ist folgendermaßen:

```
started.
finished.
```

Sie *können* den `position`-Wert unmittelbar nach dem Aufruf einer read- oder write-Methode angeben (oder zu einem beliebigen anderen Zeitpunkt) und die nächste read- oder write-Operation wird ab dieser Position ausgeführt. Beachten Sie beispielsweise, dass der folgende Code die `position`-Eigenschaft unmittelbar nach dem Aufruf einer `writeBytes()`-Operation gesetzt wird, und zwar auf den betreffenden Wert (`300`), sogar nachdem die write-Operation abgeschlossen ist:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.openAsync(myFile, FileMode.UPDATE);
myFileStream.position = 4000;
trace(myFileStream.position); // 4000
myFileStream.writeBytes(myByteArray, 0, 200);
myFileStream.position = 300;
trace(myFileStream.position); // 300
```

Der Lesepuffer und die bytesAvailable-Eigenschaft

Adobe AIR 1.0 und höher

Wenn ein FileStream-Objekt mit read-Eigenschaften (eines, in dem der Parameter `fileMode` der `open()`- oder `openAsync()`-Methode auf `READ` oder `UPDATE` gesetzt wurde) geöffnet wird, speichert die Laufzeitumgebung die Daten in einem internen Puffer. Das FileStream-Objekt beginnt, die Daten in den Puffer zu lesen, sobald Sie die Datei öffnen (durch Aufruf der `open()`- oder `openAsync()`-Methode des FileStream-Objekts).

Bei einer für synchrone Operationen (mithilfe der `open()`-Methode) geöffneten Datei können Sie den `position`-Zeiger auf jede beliebige Position (innerhalb der Dateigrenzen) setzen und eine beliebige Datenmenge (innerhalb der Dateigrenzen) lesen. Dies wird im folgenden Code (der voraussetzt, dass die Datei mindestens 100 Byte umfasst) veranschaulicht:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
myFileStream.position = 10;
myFileStream.readBytes(myByteArray, 0, 20);
myFileStream.position = 89;
myFileStream.readBytes(myByteArray, 0, 10);
```

Ob eine Datei nun für synchrone oder asynchrone Operationen geöffnet wird, die read-Methoden lesen immer aus den „verfügbaren Bytes“, die durch die Eigenschaft `bytesAvailable` dargestellt werden. Beim synchronen Lesen sind stets alle Byte einer Datei verfügbar. Beim asynchronen Lesen sind die Bytes ab der in der Eigenschaft `position` angegebenen Position verfügbar, in einer Reihe asynchroner durch `progress`-Ereignisse gemeldeter Pufferfüllungen.

Bei Dateien, die für *synchrone* Operationen geöffnet wurden, ist die Eigenschaft `bytesAvailable` immer gesetzt und repräsentiert die Anzahl der Bytes von der `position`-Eigenschaft bis zum Dateiende (alle Bytes in der Datei sind stets zum Lesen verfügbar).

Bei Dateien, die für *asynchrone* Operationen geöffnet wurden, müssen Sie sicherstellen, dass der Lesepuffer genügend Daten verbraucht hat, bevor Sie eine read-Methode aufrufen. Bei einer Datei, die asynchron geöffnet wird, während die read-Operation voranschreitet, werden die Daten aus der Datei ab der zu Beginn der read-Operation festgelegten `position` in den Puffer eingefügt, und die `bytesAvailable`-Eigenschaft wird mit jedem gelesenen Byte um eins hochgezählt. Die Eigenschaft `bytesAvailable` kennzeichnet die Anzahl verfügbarer Byte ab dem Byte an der durch die Eigenschaft `position` festgelegten Position bis zum Ende des Puffers. Das FileStream-Objekt sendet in regelmäßigen Abständen ein `progress`-Ereignis.

Bei einer asynchron geöffneten Datei löst das FileStream-Objekt in regelmäßigen Abständen das `progress`-Ereignis aus, während die Daten allmählich im Lesepuffer zur Verfügung gestellt werden. Der folgende Code beispielsweise liest Daten in das ByteArray-Objekt `bytes` ein, während es in den Puffer eingelesen wird:

```
 var bytes:ByteArray = new ByteArray();
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.openAsync(myFile, FileMode.READ);

function progressHandler(event:ProgressEvent):void
{
    myFileStream.readBytes(bytes, myFileStream.position, myFileStream.bytesAvailable);
}
```

Bei einer asynchron geöffneten Datei können nur die in den Puffer eingelesenen Daten gelesen werden. Und während Sie die Daten lesen, werden Sie aus dem Lesepuffer entfernt. Bei read-Operationen müssen Sie sicherstellen, dass die Daten auch im Lesepuffer existieren, bevor Sie die read-Operation aufrufen. Der folgende Code etwa liest 8000 Byte Daten ab der Position 4000 in die Datei:

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.addEventListener(Event.COMPLETE, completed);
myFileStream.openAsync(myFile, FileMode.READ);
myFileStream.position = 4000;

var str:String = "";

function progressHandler(event:Event):void
{
    if (myFileStream.bytesAvailable > 8000 )
    {
        str += myFileStream.readMultiByte(8000, "iso-8859-1");
    }
}
```

Während einer write-Operation liest das FileStream-Objekt keine Daten in den Lespuffer. Wenn eine write-Operation abgeschlossen wird (d. h., alle Daten im write-Puffer in die Datei geschrieben wurden), startet das FileStream-Objekt einen neuen Lesepuffer (in der Annahme, dass das verbundene FileStream-Objekt mit Leserechten geöffnet wurde) und beginnt, die Daten in den Lesepuffer zu lesen, und zwar ab der durch die Eigenschaft `position` angegebenen Position. Die Eigenschaft `position` kann die Position des letzten gelesenen Bytes bezeichnen oder auch eine andere Position, wenn der Benutzer nach der write-Operation einen anderen Wert für das `position`-Objekt angibt.

Asynchrone Programmierung und Ereignisse

Adobe AIR 1.0 und höher

Wenn eine Datei asynchron geöffnet wird (mit der Methode `openAsync()`, werden Lese- und Schreibvorgänge in der Datei asynchron ausgeführt. Während die Daten in den Lesepuffer gelesen und Ausgabedaten geschrieben werden, kann gleichzeitig anderer ActionScript-Code ausgeführt werden.

Das bedeutet, dass Sie sich für Ereignisse registrieren müssen, die vom FileStream-Objekt asynchron geöffnet wurden.

Durch die Registrierung beim `progress`-Ereignis kann der Benutzer benachrichtigt werden, wenn für den Lesevorgang neue Daten zur Verfügung stehen. Hier ein Beispiel:

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.openAsync(myFile, FileMode.READ);
var str:String = "";

function progressHandler(event:ProgressEvent):void
{
    str += myFileStream.readMultiByte(myFileStream.bytesAvailable, "iso-8859-1");
}
```

Sie können alle Daten lesen, wenn Sie sich folgendermaßen beim `complete`-Ereignis registrieren:

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completed);
myFileStream.openAsync(myFile, FileMode.READ);
var str:String = "";
function completeHandler(event:Event):void
{
    str = myFileStream.readMultiByte(myFileStream.bytesAvailable, "iso-8859-1");
}
```

Ebenso wie Eingabedaten zur Ermöglichung asynchroner Lesevorgänge gepuffert werden, werden Daten, die Sie in einen asynchronen Stream schreiben, gepuffert und asynchron in die Datei geschrieben. Während Daten in eine Datei geschrieben werden, löst das FileStream-Objekt in regelmäßigen Abständen ein `OutputProgressEvent`-Objekt aus. Ein `OutputProgressEvent`-Objekt besitzt die Eigenschaft `bytesPending` , die auf die Anzahl noch nicht geschriebener Byte gesetzt ist. Um benachrichtigt zu werden, wenn der Puffer tatsächlich in die Datei geschrieben wird, können Sie sich beim `outputProgress`-Ereignis anmelden, eventuell um ein Dialogfeld mit Fortschrittsanzeige einzublenden. Doch in der Regel ist dies nicht notwendig. Insbesondere die Methode `close()` können Sie aufrufen, ohne ungeschriebene Byte berücksichtigen zu müssen. Das FileStream-Objekt wird weiterhin Daten schreiben und das `close`-Ereignis wird ausgeliefert, nachdem das letzte Byte in die Datei geschrieben ist und die zugrundeliegende Datei wird geschlossen.

Datenformate und die geeigneten read- und write-Methoden

Adobe AIR 1.0 und höher

Jede Datei stellt eine Menge an Byte auf einem Speichermedium dar. In ActionScript können die Daten aus einer Datei immer als ein Byte-Array dargestellt werden. Der folgende Code etwa liest die Daten aus einer Datei in ein ByteArray-Objekt mit dem Namen `bytes` ein:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completeHandler);
myFileStream.openAsync(myFile, FileMode.READ);
var bytes:ByteArray = new ByteArray();

function completeHandler(event:Event):void
{
    myFileStream.readBytes(bytes, 0, myFileStream.bytesAvailable);
}
```

Entsprechend schreibt der folgende Code Daten aus einem Byte-Array mit dem Namen `bytes` in eine Datei:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.WRITE);
myFileStream.writeBytes(bytes, 0, bytes.length);
```

Oft sollen jedoch die Daten gar nicht in einem ActionScript-ByteArray-Objekt gespeichert werden. Und häufig liegt die Datendatei in einem festgelegten Dateiformat vor.

Es könnte beispielsweise vorkommen, dass die Daten in der Datei in einem Textdateiformat vorliegen und Sie die Daten in einem String-Objekt darstellen wollen.

Aus diesem Grund besitzt die FileStream-Klasse read- und write-Methoden für das Lesen von Daten aus anderen Datentypen sowie für das Schreiben in andere Datentypen als in ByteArray-Objekte. Die Methode `readMultiByte()` beispielsweise ermöglicht das Lesen von Daten aus einer Datei und Speichern in einem String. Hier ein Codebeispiel:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completed);
myFileStream.openAsync(myFile, FileMode.READ);
var str:String = "";

function completeHandler(event:Event):void
{
    str = myFileStream.readMultiByte(myFileStream.bytesAvailable, "iso-8859-1");
}
```

Der zweite Parameter der `readMultiByte()`-Methode gibt den Zeichensatz an (im Beispiel „iso-8859-1“), in dem ActionScript die Daten interpretieren soll. Adobe AIR unterstützt gängige Zeichensatzkodierungen (siehe Unterstützte Zeichensätze).

Die FileStream-Klasse besitzt auch die Methode `readUTFBytes()`, die unter Verwendung des UTF-8-Zeichensatzes Daten aus dem Lesepuffer in einen String liest. Zeichen im UTF-8-Zeichensatz haben unterschiedliche binäre Länge. Sie sollten also `readUTFBytes()` nicht in einer Methode verwenden, die auf das `progress`-Ereignis reagiert, da die Daten am Ende des Lesepuffers ein unvollständiges Zeichen repräsentierten könnten. (Dies trifft auch zu, wenn die Methode `readMultiByte()` mit einer anderen Kodierungsform variabler Länge verwendet wird.) Aus diesem Grund sollten Sie die gesamte Datenmenge lesen, wenn das FileStream-Objekt das `complete`-Ereignis auslöst.

Es gibt auch ähnliche write-Methoden für die Arbeit mit String-Objekten und Textdateien, `writeMultiByte()` und `writeUTFBytes()`.

Die Methoden `readUTF()` und `writeUTF()` (nicht zu verwechseln mit `readUTFBytes()` und `writeUTFBytes()`) lesen und schreiben ebenfalls Textdaten aus und in eine Datei, setzen aber voraus, dass den Textdaten Daten vorangestellt sind, die die Länge der Textdaten angeben, was in Standardtextdateien nicht unbedingt üblich ist.

Einige UTF-kodierte Textdateien beginnen mit einem „UTF-BOM" (Byte Order Mark), einer Marke für die Bytereihenfolge, welche sowohl das Kodierungsschema (Little oder Big Endian, LE oder BE) als auch die Kodierungsform (z. B. UTF-16 oder UTF-32) definiert.

Ein Beispiel für das Lesen aus einer und Schreiben in eine Textdatei finden Sie unter „Beispiel: Einlesen einer XML-Datei in ein XML-Objekt" auf Seite 744.

Das `readObject()` und das `writeObject()` sind praktische Möglichkeiten, um Daten für komplexe ActionScript-Objekte zu speichern und abzurufen. Die Daten werden im AMF (ActionScript Message Format) kodiert. Adobe AIR, Flash Player, Flash Media Server und Flex Data Services enthalten APIs für die Arbeit mit Daten in diesem Format.

Es gibt noch einige weitere read- und write-Methoden (etwa `readDouble()` und `writeDouble()`). Wenn Sie diese Methoden jedoch verwenden, sollten Sie sicherstellen, dass das Dateiformat dem von diesen Methoden definierten Format entspricht.

Dateiformate sind häufig komplexer als einfache Textformate. Eine MP3-Datei beispielsweise umfasst komprimierte Daten, die nur mit für MP3-Dateien spezifischen Dekomprimierungs- und Dekodierungsalgorithmen interpretiert werden können. MP3-Dateien können auch ID3-Tags mit Metatag-Informationen über die Datei enthalten (z. B. Titel und Interpret eines Musikstückes). Es gibt mehrere Versionen des ID3-Formats, von denen die einfachste (ID3 Version 1) im Abschnitt „Beispiel: Lesen aus und Schreiben in zufällige Stellen einer Datei" auf Seite 745 erläutert wird.

Andere Dateiformate (für Bilder, Datenbanken, Anwendungsdokumente usw.) haben wiederum andere Strukturen und wenn Sie in ActionScript mit ihren Daten arbeiten wollen, müssen Sie die Struktur dieser Daten verstehen.

Verwenden der load()- und save()-Methoden

Flash Player 10 und höher, Adobe AIR 1.5 und höher

In Flash Player 10 wurden die `load()`- und `save()`-Methoden zur FileReference-Klasse hinzugefügt. Diese Methoden gibt es auch in AIR 1.5; und die File-Klasse übernimmt die Methoden von der FileReference-Klasse. Diese Methoden wurden entwickelt, um in Flash Player eine sichere Möglichkeit zum Laden und Speichern von Dateien bereitzustellen. AIR-Anwendungen können diese Methoden jedoch auch als einfache Möglichkeit zum asynchronen Laden und Speichern von Dateien verwenden.

Im folgenden Beispiel wird ein String in einer Textdatei gespeichert:

```
var file:File = File.applicationStorageDirectory.resolvePath("test.txt");
var str:String = "Hello.";
file.addEventListener(Event.COMPLETE, fileSaved);
file.save(str);
function fileSaved(event:Event):void
{
    trace("Done.");
}
```

Der `data`-Parameter der `save()`-Methode kann einen String-, XML- oder ByteArray-Wert aufweisen. Wenn das Argument ein String- oder XML-Wert ist, speichert die Methode die Datei als UTF-8-kodierte Textdatei.

Wenn dieses Codebeispiel ausgeführt wird, zeigt die Anwendung ein Dialogfeld an, in dem der Benutzer ein Ziel für die gespeicherte Datei auswählt.

Im folgenden Beispiel wird ein String aus einer UTF-8-kodierten Textdatei geladen:

```
var file:File = File.applicationStorageDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, loaded);
file.load();
var str:String;
function loaded(event:Event):void
{
    var bytes:ByteArray = file.data;
    str = bytes.readUTFBytes(bytes.length);
    trace(str);
}
```

Die FileStream-Klasse bietet mehr Funktionen als die von den `load()`- und `save()`-Methoden bereitgestellten:

- Mithilfe der FileStream-Klasse können Sie Daten sowohl synchron als auch asynchron lesen und schreiben.
- Mithilfe der FileStream-Klasse können Sie inkrementell in eine Datei schreiben.
- Mithilfe der FileStream-Klasse können Sie eine Datei für den zufälligen Zugriff öffnen (lesen aus und schreiben in beliebige Abschnitte der Datei).
- Mithilfe der FileStream-Klasse können Sie angeben, welche Art Zugriff Sie auf die Datei haben, indem Sie den `fileMode`-Parameter der `open()`- oder `openAsync()`-Methode festlegen.
- Mithilfe der FileStream-Klasse können Sie Daten in Dateien speichern, ohne dass dem Benutzer ein Dialogfeld zum Öffnen oder Speichern angezeigt wird.
- Sie können andere Typen als Byte-Arrays direkt verwenden, wenn Sie Daten mit der FileStream-Klasse lesen.

Beispiel: Einlesen einer XML-Datei in ein XML-Objekt

Adobe AIR 1.0 und höher

Die folgenden Beispiele verdeutlichen Lese- und Schreibvorgänge in XML-Dateien.

Um aus einer XML-Datei zu lesen, initialisieren Sie das File- und das FileStream-Objekt, rufen die Methode `readUTFBytes()` des FileStream-Objekts auf und konvertieren den String in ein XML-Objekt:

```
 var file:File = File.documentsDirectory.resolvePath("AIR Test/preferences.xml");
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.READ);
var prefsXML:XML = XML(fileStream.readUTFBytes(fileStream.bytesAvailable));
fileStream.close();
```

Entsprechend einfach ist das Schreiben der Daten in die XML-Datei: Sie richten das entsprechende File- und FileStream-Objekt ein und rufen dann eine write-Methode des FileStream-Objekts auf. Übergeben Sie dann, wie im folgenden Code, die String-Version der XML-Daten an die write-Methode:

```
var prefsXML:XML = <prefs><autoSave>true</autoSave></prefs>;
var file:File = File.documentsDirectory.resolvePath("AIR Test/preferences.xml");
fileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);

var outputString:String = '<?xml version="1.0" encoding="utf-8"?>\n';
outputString += prefsXML.toXMLString();

fileStream.writeUTFBytes(outputString);
fileStream.close();
```

Die folgenden Beispiele verwenden die Methoden `readUTFBytes()` und `writeUTFBytes()`, da sie davon ausgehen, dass die Daten im UTF-8-Format gespeichert wurden. Wenn nicht, müssen Sie möglicherweise eine andere Methode verwenden (siehe „Datenformate und die geeigneten read- und write-Methoden“ auf Seite 742).

Das vorige Beispiel verwendet FileStream-Objekte, die für synchrone Operationen geöffnet wurden. Sie können auch Dateien für asynchrone Operationen öffnen (die sich auf Ereignis-Listener-Funktionen verlassen, die auf bestimmte Ereignisse reagieren). Der folgende Code beispielsweise zeigt, wie eine XML-Datei asynchron gelesen wird:

```
var file:File = File.documentsDirectory.resolvePath("AIR Test/preferences.xml");
var fileStream:FileStream = new FileStream();
fileStream.addEventListener(Event.COMPLETE, processXMLData);
fileStream.openAsync(file, FileMode.READ);
var prefsXML:XML;

function processXMLData(event:Event):void
{
    prefsXML = XML(fileStream.readUTFBytes(fileStream.bytesAvailable));
    fileStream.close();
}
```

Die Methode `processXMLData()` wird aufgerufen, wenn die gesamte Datei in den Lesepuffer gelesen wird (wenn das FileStream-Objekt das `complete`-Ereignis auslöst). Sie ruft die Methode `readUTFBytes()` auf, um eine Stringversion der Lesedaten abzurufen, und erstellt aufgrund dieses Strings das XML-Objekt `prefsXML`.

Eine Beispielanwendung mit diesen Fähigkeiten finden Sie unter Lesen und Schreiben von einer XML-Einstellungsdatei.

Beispiel: Lesen aus und Schreiben in zufällige Stellen einer Datei

Adobe AIR 1.0 und höher

MP3-Dateien können auch ID3-Tags enthalten. Das sind Abschnitte am Anfang oder Ende der Datei, die Metadaten für die Identifikation der Aufnahme enthalten. Von dem ID3-Tag-Format selbst gibt es verschiedene Versionen. Das folgenden Beispiel beschreibt, wie aus einer MP3-Datei gelesen bzw. in eine MP3-Datei geschrieben wird, die das einfachste ID3-Format enthält (ID3, Version 1.0). Dies geschieht durch einen „Zufallszugriff auf die Dateidaten“, d. h., es wird aus zufälligen Stellen in der Datei gelesen bzw. in zufällige Stellen geschrieben.

Bei einer MP3-Datei, die ein ID3-Tag der Version 1 enthält, befinden sich die ID3-Daten am Ende der Datei, d. h., in den letzten 128 Byte.

Beim Zugriff auf eine Datei mit zufälligem Schreib-/Lesezugriff ist es wichtig, `FileMode.UPDATE` als `fileMode`-Parameter für die `open()`- oder `openAsync()`-Methode anzugeben:

```
var file:File = File.documentsDirectory.resolvePath("My Music/Sample ID3 v1.mp3");
var fileStr:FileStream = new FileStream();
fileStr.open(file, FileMode.UPDATE);
```

Dadurch können Sie aus der Datei lesen wie auch in die Datei schreiben.

Beim Öffnen der Datei können Sie den `position`-Zeiger auf die Position „128 Byte vor Dateiende“ setzen:

```
fileStr.position = file.size - 128;
```

Dieser Code setzt die Eigenschaft `position` auf diese Stelle in der Datei, da das ID3-Format der Version 1.0 festlegt, dass die ID3-Tag-Daten in den letzten 128 Byte einer Datei gespeichert werden. Darüber hinaus hat die Spezifikation die folgenden Vorschriften:

- Die ersten drei Byte des Tags enthalten den String „`TAG`".
- Die nächsten 30 Zeichen enthalten den Titel für die MP3-Spur, und zwar als String.
- Die nächsten 30 Zeichen enthalten den Namen des Interpreten, ebenfalls als String.
- Die nächsten 30 Zeichen enthalten den Namen des Albums als String.
- Die nächsten 4 Zeichen enthalten das Erscheinungsjahr als String.
- Die nächsten 30 Zeichen enthalten den Kommentar als String.
- Das nächste Byte enthält einen Code, der das Genre dieses Titels bezeichnet.
- Alle Textdaten haben das ISO 8859-1-Format.

Die Methode `id3TagRead()` überprüft die Daten, nachdem sie gelesen wurden (nach dem `complete`-Ereignis):

```
function id3TagRead():void
{
    if (fileStr.readMultiByte(3, "iso-8859-1").match(/tag/i))
    {
        var id3Title:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3Artist:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3Album:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3Year:String = fileStr.readMultiByte(4, "iso-8859-1");
        var id3Comment:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3GenreCode:String =  fileStr.readByte().toString(10);
    }
}
```

Sie können auch mit einem Zufallszugriff in die Datei schreiben. Sie könnten etwa die Variable `id3Title` mit einem Parser analysieren, um sicherzustellen, dass die Groß-/Kleinschreibung stimmt (anhand von Methoden der String-Klasse), und dann einen geänderten String mit dem Namen `newTitle` in die Datei zurückschreiben. Hier der Code:

```
fileStr.position = file.length - 125;    // 128 - 3
fileStr.writeMultiByte(newTitle, "iso-8859-1");
```

Damit der `newTitle`-String dem ID3-Standard der Version 1 entspricht, muss die Länge 30 Zeichen betragen und am Ende muss der Zeichencode 0 (`String.fromCharCode(0)`) stehen.

Kapitel 39: Speichern lokaler Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie verwenden die SharedObject-Klasse zum Speichern von kleinen Datenmengen auf dem Client. In Adobe AIR können Sie auch die EncryptedLocalStore-Klasse verwenden, um kleine Mengen vertraulicher Benutzerdaten in einer AIR-Anwendung auf dem lokalen Computer zu speichern.

Sie können auch Dateien im Dateisystem lesen und schreiben und (in Adobe AIR) auf lokale Datenbankdateien zugreifen. Weitere Informationen finden Sie unter „Arbeiten mit dem Dateisystem“ auf Seite 695 und „Arbeiten mit lokalen SQL-Datenbanken in AIR“ auf Seite 761.

Es gibt eine Reihe von Sicherheitsfaktoren, die mit gemeinsam genutzten Objekten zu tun haben. Weitere Informationen finden Sie unter „Gemeinsame Objekte“ auf Seite 1151 in „Sicherheit“ auf Seite 1115.

Gemeinsame Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein gemeinsames Objekt, das gelegentlich auch als „Flash-Cookie“ bezeichnet wird, ist eine Datendatei, die von den von Ihnen besuchten Sites auf Ihrem Computer erstellt werden kann. Gemeinsame Objekte werden auch häufig zum Aufwerten Ihres Browsing-Erlebnisses verwendet, z. B. ermöglichen gemeinsame Objekte, das Erscheinungsbild einer häufig besuchten Website zu personalisieren.

Gemeinsame Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Gemeinsame Objekte funktionieren wie Cookies. Mit der SharedObject-Klasse können Sie Daten auf der lokalen Festplatte des Benutzers speichern und diese Daten während derselben Sitzung oder in einer späteren Sitzung aufrufen. Anwendungen können nur auf ihre eigenen SharedObject-Daten zugreifen und dies nur, wenn sie auf derselben Domäne ausgeführt werden. Die Daten werden nicht an den Server geschickt und sind für andere Anwendungen, die in anderen Domänen ausgeführt werden, nicht zugänglich. Sie können jedoch den Anwendungen in derselben Domäne den Zugang ermöglichen.

Gemeinsame Objekte im Vergleich zu Cookies

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Cookies und gemeinsame Objekte sind sehr ähnlich. Da viele Programmierer wissen, wie Cookies funktionieren, ist es angebracht, Cookies mit lokalen SharedObjects zu vergleichen.

Cookies, die dem RFC 2109-Standard entsprechen, weisen in der Regel nachstehende Eigenschaften auf:

- Sie laufen ab und zwar oft am Ende der Sitzung.
- Sie können vom Client für eine bestimmte Site deaktiviert werden.
- Es können maximal 300 Cookies insgesamt und 20 Cookies pro Site gespeichert werden.

- Im Allgemeinen ist ihre Größe auf jeweils 4 KB beschränkt.
- Manchmal werden sie als Sicherheitsbedrohung empfunden und aus diesem Grund auf dem Client deaktiviert.
- Sie werden an einem vom Clientbrowser bestimmten Ort gespeichert.
- Sie werden vom Client über HTTP an den Server übermittelt.

  Gemeinsame Objekte haben hingegen die nachstehenden Eigenschaften:

- Standardmäßig laufen sie nicht ab.
- Standardmäßig ist ihre Größe auf jeweils 100 KB beschränkt.
- Sie können einfache Daten speichern (wie z. B. String, Array und Datum)
- Sie werden an einem von der Anwendung bestimmten Ort (im Stammverzeichnis des Benutzers) gespeichert.
- Sie werden nie vom Client an den Server übertragen.

Die SharedObject-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der SharedObject-Klasse können Sie gemeinsame Objekte erstellen und löschen und die aktuelle Größe eines verwendeten SharedObject-Objekts ermitteln.

Erstellen eines gemeinsamen Objekts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Erstellen eines SharedObject-Objekts verwenden Sie die `SharedObject.getLocal()`-Methode. Für diese Methode gilt folgende Syntax:

```
SharedObject.getLocal("objectName" [, pathname]): SharedObject
```

Im nachstehenden Beispiel wird ein gemeinsames Objekt mit dem Namen mySO erstellt:

```
public var mySO:SharedObject;
mySO = SharedObject.getLocal("preferences");
```

Dies erstellt auf dem Client eine Datei mit dem Namen „preferences.sol“.

Der Begriff *lokal* bezieht sich auf den Speicherort des gemeinsamen Objekts. In diesem Fall speichert Adobe®Flash® Player die SharedObject-Datei lokal im Stammverzeichnis des Clients.

Wenn Sie ein gemeinsames Objekt erstellen, legt Flash Player in der Sandbox einen neuen Ordner für die Anwendung und die Domäne an. Flash Player erstellt auch eine *.sol-Datei, in der die SharedObject-Daten gespeichert werden. Der Standardspeicherort dieser Datei ist ein Unterverzeichnis des Stammverzeichnisses des Benutzers. In der nachstehenden Tabelle sind die Standardspeicherorte dieses Verzeichnisses aufgeführt:

Betriebssystem	Verzeichnis
Windows 95/98/ME/2000/XP	`c:/Documents and Settings/`*`username`*`/Application Data/Macromedia/Flash Player/#SharedObjects`

Betriebssystem	Verzeichnis
Windows Vista/Windows 7	`c:/Users/`*`username`*`/AppData/Roaming/Macromedia/Flash Player/#SharedObjects`
Macintosh OS X	`/Users/`*`username`*`/Library/Preferences/Macromedia/Flash Player/#SharedObjects/`*`web_domain/path_to_application/application_name/object_name`*`.sol`
Linux/Unix	`/home/username/.macromedia/Flash_Player/#SharedObjects/`*`web_domain/path_to_application/application_name/object_name`*`.sol`

Unter dem Verzeichnis #SharedObjects befindet sich ein nach dem Zufallsprinzip benanntes Verzeichnis. Darunter liegt ein weiteres Verzeichnis, das den Hostnamen, den Pfad der Anwendung und schließlich den Namen der *.sol-Datei enthält.

Wenn Sie zum Beispiel eine Anwendung mit dem Namen MyApp.swf auf dem lokalen Host in einem Unterverzeichnis namens /sos anfordern, legt Flash Player die *.sol-Datei in Windows XP am folgenden Speicherort ab:

```
c:/Documents and Settings/fred/Application Data/Macromedia/Flash
Player/#SharedObjects/KROKWXRK/#localhost/sos/MyApp.swf/data.sol
```

Hinweis: *Geben Sie keinen Namen in der `SharedObject.getLocal()`-Methode an, benennt Flash Player die Datei als „undefined.sol“.*

Standardmäßig kann Flash permanente SharedObject-Objekte von bis zu 100 KB pro Domäne lokal speichern. Der Benutzer kann diesen Wert ändern. Wenn die Anwendung versucht, Daten in einem gemeinsamen Objekt zu speichern, und dabei die 100-KB-Grenze überschritten wird, zeigt Flash Player das Dialogfeld zum lokalen Speichern an. Hier kann der Benutzer die Zuteilung von mehr Speicherplatz für die anfordernde Domäne genehmigen oder verweigern.

Angeben eines Pfads

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Über den optionalen *pathname*-Parameter können Sie den Speicherort der SharedObject-Datei festlegen. Diese Datei muss ein Unterverzeichnis des SharedObject-Verzeichnisses der Domäne sein. Wenn Sie zum Beispiel eine Anwendung auf dem Localhost anfordern und Folgendes angeben:

```
mySO = SharedObject.getLocal("myObjectFile","/");
```

schreibt Flash Player die SharedObject-Datei in das Verzeichnis /#localhost (bzw. /localhost für Offline-Anwendungen). Dies ist nützlich, wenn mehrere Anwendungen auf dem Client auf dasselbe gemeinsame Objekt zugreifen sollen. In diesem Fall kann der Client zwei Flex-Anwendungen ausführen, die beide den Pfad zum gemeinsamen Objekt angeben, das das Stammverzeichnis der Domäne ist; der Client kann dann über beide Anwendungen auf dasselbe gemeinsame Objekt zugreifen. Sie können das LocalConnection-Objekt verwenden, damit mehrere Anwendungen auf nicht permanente Weise auf gemeinsame Daten zugreifen können.

Wenn Sie ein Verzeichnis angeben, das nicht existiert, erstellt Flash Player keine SharedObject-Datei.

Hinzufügen von Daten zu einem gemeinsamen Objekt

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Hinzufügen von Daten zu der *.sol-Datei eines SharedObject-Objekts verwenden Sie die `data`-Eigenschaft des SharedObject-Objekts. Mithilfe der nachstehenden Syntax fügen Sie neue Daten zum gemeinsamen Objekt hinzu:

```
sharedObject_name.data.variable = value;
```

Im nachstehenden Beispiel werden die Eigenschaften `userName`, `itemNumbers` und `adminPrivileges` und deren Werte zu einem SharedObject hinzugefügt:

```
public var currentUserName:String = "Reiner";
public var itemsArray:Array = new Array(101,346,483);
public var currentUserIsAdmin:Boolean = true;
mySO.data.userName = currentUserName;
mySO.data.itemNumbers = itemsArray;
mySO.data.adminPrivileges = currentUserIsAdmin;
```

Nachdem Sie der `data`-Eigenschaft Werte zugewiesen haben, müssen Sie den Flash Player anweisen, diese Werte in die SharedObject-Datei zu schreiben. Damit Flash Player die Werte zwingend in die SharedObject-Datei schreibt, verwenden Sie die `SharedObject.flush()`-Methode, wie nachstehend beschrieben:

```
mySO.flush();
```

Wenn Sie die `SharedObject.flush()`-Methode nicht aufrufen, schreibt Flash Player die Werte in die Datei, sobald die Anwendung beendet wird. So kann der Benutzer jedoch nicht den Datenspeicherplatz für Flash Player vergrößern, wenn die Datenmenge die Standardeinstellungen überschreitet. Deshalb ist es ratsam, `SharedObject.flush()` aufzurufen.

Wenn Sie die `flush()`-Methode zum Schreiben von gemeinsamen Objekten auf die Festplatte eines Benutzers verwenden, müssen Sie prüfen, ob der Benutzer die lokale Speicherung mit dem Einstellungsmanager von Flash Player (www.macromedia.com/support/documentation/de/flashplayer/help/settings_manager07.html) explizit deaktiviert hat. Ein Beispiel hierfür wird im folgenden Code gezeigt:

```
var so:SharedObject = SharedObject.getLocal("test");
trace("Current SharedObject size is " + so.size + " bytes.");
so.flush();
```

Speichern von Objekten in gemeinsamen Objekten

Sie können einfache Objekte, wie Arrays oder Strings, in der `data`-Eigenschaft eines gemeinsamen Objekts speichern.

Das nachstehende Beispiel ist eine ActionScript-Klasse zur Bestimmung der Methoden, die für die Steuerung der Interaktion mit dem gemeinsamen Objekt verwendet werden. Mithilfe dieser Methoden kann der Benutzer Objekte zum gemeinsamen Objekt hinzuzufügen oder aus diesem entfernen. In dieser Klasse wird eine ArrayCollection gespeichert, die einfache Objekte enthält.

```
package {
    import mx.collections.ArrayCollection;
    import flash.net.SharedObject;

    public class LSOHandler {

        private var mySO:SharedObject;
        private var ac:ArrayCollection;
        private var lsoType:String;

        // The parameter is "feeds" or "sites".
        public function LSOHandler(s:String) {
            init(s);
        }

        private function init(s:String):void {
            ac = new ArrayCollection();
            lsoType = s;
            mySO = SharedObject.getLocal(lsoType);
            if (getObjects()) {
                ac = getObjects();
            }
        }

        public function getObjects():ArrayCollection {
            return mySO.data[lsoType];
        }

        public function addObject(o:Object):void {
            ac.addItem(o);
            updateSharedObjects();
        }

        private function updateSharedObjects():void {
            mySO.data[lsoType] = ac;
            mySO.flush();
        }
    }

}
```

Die folgende Flex-Anwendung erstellt eine Instanz der ActionScript-Klasse für jeden Typ gemeinsamer Objekte, die sie braucht. Anschließend ruft sie Methoden dieser Klasse auf, wenn der Benutzer Blogs oder Site-URLs hinzufügt oder entfernt.

```
<?xml version="1.0"?>
<!-- lsos/BlogAggregator.mxml -->
<mx:Application
    xmlns:local="*"
    xmlns:mx="http://www.adobe.com/2006/mxml"
    creationComplete="initApp()"
    backgroundColor="#ffffff"
>
    <mx:Script>
        <![CDATA[
        import mx.collections.ArrayCollection;
        import mx.utils.ObjectUtil;
        import flash.net.SharedObject;

        [Bindable]
        public var welcomeMessage:String;

        [Bindable]
        public var localFeeds:ArrayCollection = new ArrayCollection();

        [Bindable]
        public var localSites:ArrayCollection = new ArrayCollection();

        public var lsofeeds:LSOHandler;
        public var lsosites:LSOHandler;

        private function initApp():void {
            lsofeeds = new LSOHandler("feeds");
            lsosites = new LSOHandler("sites");

            if (lsofeeds.getObjects()) {
                localFeeds = lsofeeds.getObjects();
            }
            if (lsosites.getObjects()) {
                localSites = lsosites.getObjects();
            }
        }

        // Adds a new feed to the feeds DataGrid.
        private function addFeed():void {
            // Construct an object you want to store in the
            // LSO. This object can contain any number of fields.
            var o:Object = {name:ti1.text, url:ti2.text, date:new Date()};
            lsofeeds.addObject(o);

            // Because the DataGrid's dataProvider property is
            // bound to the ArrayCollection, Flex updates the
            // DataGrid when you call this method.
            localFeeds = lsofeeds.getObjects();

            // Clear the text fields.
            ti1.text = '';
            ti2.text = '';
        }

        // Removes feeds from the feeds DataGrid.
        private function removeFeed():void {
```

```
            // Use a method of ArrayCollection to remove a feed.
            // Because the DataGrid's dataProvider property is
            // bound to the ArrayCollection, Flex updates the
            // DataGrid when you call this method. You do not need
            // to update it manually.
            if (myFeedsGrid.selectedIndex > -1) {

localFeeds.removeItemAt(myFeedsGrid.selectedIndex);
             }
        }

        private function addSite():void {
            var o:Object = {name:ti3.text, date:new Date()};
            lsosites.addObject(o);
            localSites = lsosites.getObjects();
            ti3.text = '';
        }

        private function removeSite():void {
            if (mySitesGrid.selectedIndex > -1) {

localSites.removeItemAt(mySitesGrid.selectedIndex);
            }
        }

        ]]>
    </mx:Script>

    <mx:Label text="Blog aggregator" fontSize="28"/>

    <mx:Panel title="Blogs">
        <mx:Form id="blogForm">
            <mx:HBox>
                <mx:FormItem label="Name:">
                    <mx:TextInput id="ti1" width="100"/>
                </mx:FormItem>
                <mx:FormItem label="Location:">
                    <mx:TextInput id="ti2" width="300"/>
                </mx:FormItem>
                <mx:Button id="b1" label="Add Feed" click="addFeed()"/>
            </mx:HBox>

            <mx:FormItem label="Existing Feeds:">
                <mx:DataGrid
                    id="myFeedsGrid"
                    dataProvider="{localFeeds}"
                    width="400"
                />
            </mx:FormItem>
            <mx:Button id="b2" label="Remove Feed" click="removeFeed()"/>
        </mx:Form>
    </mx:Panel>

    <mx:Panel title="Sites">
```

```
        <mx:Form id="siteForm">
            <mx:HBox>
                <mx:FormItem label="Site:">
                    <mx:TextInput id="ti3" width="400"/>
                </mx:FormItem>
                <mx:Button id="b3" label="Add Site" click="addSite()"/>
            </mx:HBox>

            <mx:FormItem label="Existing Sites:">
                <mx:DataGrid
                    id="mySitesGrid"
                    dataProvider="{localSites}"
                    width="400"
                />
            </mx:FormItem>
            <mx:Button id="b4" label="Remove Site" click="removeSite()"/>
        </mx:Form>
    </mx:Panel>

</mx:Application>
```

Speichern von typisierten Objekten in gemeinsamen Objekten

Sie können typisierte ActionScript-Instanzen in gemeinsamen Objekten speichern. Rufen Sie dazu die `flash.net.registerClassAlias()`-Methode auf, um die Klasse zu registrieren. Sie erhalten eine typisierte Instanz, wenn Sie eine Instanz der Klasse erstellen und sie im Datenmember des gemeinsamen Objekts speichern und später das Objekt lesen. Standardmäßig unterstützt die `objectEncoding`-Eigenschaft des SharedObject die AMF3-Kodierung und extrahiert die gespeicherte Instanz aus dem SharedObject-Objekt; der Typ der gespeicherten Instanz entspricht jenem, den Sie beim Aufruf der `registerClassAlias()`-Methode angegeben haben.

(nur iOS) Cloud-Backup für lokale freigegebene Objekte verhindern

Adobe AIR 3.7 und höher – nur iOS

Sie können die `SharedObject.preventBackup`-Eigenschaft festlegen, um zu steuern, ob lokale freigegebene Objekte über den iOS-Cloud-Backup-Dienst gesichert werden. Dies ist laut Apple erforderlich bei Inhalten, die neu generiert oder heruntergeladen werden können, jedoch für das einwandfreie Funktionieren Ihrer Anwendung bei Offline-Verwendung notwendig sind.

Erstellen von mehreren gemeinsamen Objekten

Sie können mehrere gemeinsame Objekte für dieselbe Flex-Anwendung erstellen. Weisen Sie dazu jedem einen anderen Instanznamen zu, wie im nachstehenden Beispiel dargestellt:

```
public var mySO:SharedObject = SharedObject.getLocal("preferences");
public var mySO2:SharedObject = SharedObject.getLocal("history");
```

So werden die Dateien „preferences.sol“ und „history.sol“ im Stammverzeichnis der Flex-Anwendung angelegt.

Erstellen eines sicheren SharedObject

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie mit `getLocal()` oder `getRemote()` ein lokales oder entferntes SharedObject erstellen, gibt es einen optionalen Parameter namens `secure`, mit dem festgelegt werden kann, ob der Zugriff auf dieses gemeinsame Objekt auf SWF-Dateien beschränkt ist, die über eine HTTPS-Verbindung bereitgestellt werden. Wenn dieser Parameter auf `true` eingestellt ist und Ihre SWF-Datei über HTTPS bereitgestellt wird, erstellt Flash Player ein neues sicheres gemeinsames Objekt oder ruft einen Verweis auf ein vorhandenes sicheres gemeinsames Objekt ab. Dieses sichere gemeinsame Objekt kann nur durch SWF-Dateien gelesen oder geschrieben werden, die über HTTPS bereitgestellt werden und die `SharedObject.getLocal()` mit der Einstellung `true` für den secure-Parameter aufrufen. Wenn dieser Parameter auf `false` eingestellt ist und Ihre SWF-Datei über HTTPS bereitgestellt wird, erstellt Flash Player ein neues gemeinsames Objekt oder ruft einen Verweis auf ein vorhandenes gemeinsames Objekt ab.

An diesem gemeinsamen Objekt können nur SWF-Dateien Lese- oder Schreibvorgänge durchführen, die über andere Verbindungen als HTTPS bereitgestellt werden. Wenn die SWF-Datei über eine andere Verbindung als HTTPS bereitgestellt wird und Sie diesen Parameter auf `true` einstellen, schlägt die Erstellung des neuen gemeinsamen Objekts (bzw. der Zugriff auf ein zuvor erstelltes sicheres gemeinsames Objekt) fehl. Es wird ein Fehler ausgelöst und das gemeinsame Objekt wird auf `null` gesetzt. Wenn Sie versuchen, den folgenden Codeausschnitt über eine andere Verbindung als HTTPS auszuführen, löst die `SharedObject.getLocal()`-Methode einen Fehler aus:

```
try
{
    var so:SharedObject = SharedObject.getLocal("contactManager", null, true);
}
catch (error:Error)
{
    trace("Unable to create SharedObject.");
}
```

Unabhängig vom Wert dieses Parameters belegen die erstellten gemeinsamen Objekte einen Teil des von einer Domäne belegten Speicherplatzes.

Anzeigen des Inhalts eines gemeinsamen Objekts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Werte werden in der `data`-Eigenschaft von gemeinsamen Objekten gespeichert. Sie können jeden Wert innerhalb einer gemeinsamen Objektinstanz durchlaufen. Verwenden Sie hierzu eine `for..in`-Schleife, wie im folgenden Beispiel gezeigt:

```
var so:SharedObject = SharedObject.getLocal("test");
so.data.hello = "world";
so.data.foo = "bar";
so.data.timezone = new Date().timezoneOffset;
for (var i:String in so.data)
{
    trace(i + ":\t" + so.data[i]);
}
```

Zerstören von gemeinsamen Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie zerstören ein SharedObject auf dem Client, indem Sie die `SharedObject.clear()`-Methode verwenden. Die Verzeichnisse im Standardpfad der gemeinsamen Objekte der Anwendung werden dabei nicht zerstört.

Im folgenden Beispiel wird die SharedObject-Datei auf dem Client gelöscht:

```
public function destroySharedObject():void {
    mySO.clear();
}
```

SharedObject-Beispiel

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das folgende Beispiel zeigt, dass Sie einfache Objekte, wie beispielsweise Date-Objekte, in einem SharedObject-Objekt speichern können, ohne diese Objekte manuell zu serialisieren oder zu deserialisieren.

Im nachstehenden Beispiel werden Sie zunächst als neuer Besucher begrüßt. Wenn Sie auf „Abmelden“ klicken, speichert die Anwendung das aktuelle Datum in einem gemeinsamen Objekt. Wenn Sie diese Anwendung das nächste Mal starten oder die Seite aktualisieren, gibt die Anwendung die Uhrzeit Ihrer Abmeldung an.

Um zu sehen, wie die Anwendung funktioniert, starten Sie die Anwendung, klicken auf „Abmelden“ und aktualisieren anschließend die Seite. Die Anwendung zeigt das Datum und die Uhrzeit der Abmeldung bei Ihrem letzten Besuch. Sie können die gespeicherten Informationen jederzeit löschen. Klicken Sie dazu auf die Schaltfläche „Delete LSO“.

```
<?xml version="1.0"?>
<!-- lsos/WelcomeMessage.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" initialize="initApp()">
  <mx:Script><![CDATA[
  public var mySO:SharedObject;
  [Bindable]
  public var welcomeMessage:String;

  public function initApp():void {
     mySO = SharedObject.getLocal("mydata");
     if (mySO.data.visitDate==null) {
        welcomeMessage = "Hello first-timer!"
     } else {
        welcomeMessage = "Welcome back. You last visited on " +
           getVisitDate();
     }
  }

  private function getVisitDate():Date {
     return mySO.data.visitDate;
  }

  private function storeDate():void {
     mySO.data.visitDate = new Date();
     mySO.flush();
  }

  private function deleteLSO():void {
     // Deletes the SharedObject from the client machine.
     // Next time they log in, they will be a 'first-timer'.
     mySO.clear();
  }

  ]]></mx:Script>
  <mx:Label id="label1" text="{welcomeMessage}"/>
  <mx:Button label="Log Out" click="storeDate()"/>
  <mx:Button label="Delete LSO" click="deleteLSO()"/>
</mx:Application>
```

Verschlüsselter lokaler Speicher

Die EncryptedLocalStore-Klasse (ELS) stellt einen Mechanismus für verschlüsselten lokalen Speicher bereit, den Sie als kleinen Cache für die privaten Daten einer Anwendung verwenden können. ELS-Daten können nicht von Anwendungen gemeinsam genutzt werden. ELS hat den Zweck, einer Anwendung das Speichern von leicht wiederherzustellenden Elementen wie Anmeldeinformationen und anderen privaten Informationen zu ermöglichen. ELS-Daten sollten nicht als dauerhaft betrachtet werden; siehe „Einschränkungen des verschlüsselten lokalen Speichers“ und „Empfohlene Verfahren“ weiter unten.

Hinweis: *Zusätzlich zum verschlüsselten lokalen Speicher ermöglicht AIR auch die Verschlüsselung von Inhalten, die in SQL-Datenbanken gespeichert sind. Weitere Informationen finden Sie unter „Verwenden von Verschlüsselung mit SQL-Datenbanken“ auf Seite 807.*

Es empfiehlt sich, den verschlüsselten lokalen Speicher zum Zwischenspeichern von Daten zu verwenden, die gesichert werden müssen (z. B. Anmeldeinformationen für Webdienste). Der ELS eignet sich für die Speicherung von Informationen, die für andere Benutzer nicht zugänglich sein sollen. Er schützt die Daten jedoch nicht vor anderen Prozessen, die unter demselben Benutzerkonto ausgeführt werden. ELS eignet sich deshalb nicht für den Schutz geheimer Anwendungsdaten wie DRM oder Verschlüsselungsschlüssel.

Auf Desktopplattformen verwendet AIR DPAPI unter Windows, KeyChain unter Mac OS und iOS sowie KeyRing oder KWallet unter Linux, um den verschlüsselten lokalen Speicher den einzelnen Anwendungen und Benutzern zuzuordnen. Die Verschlüsselung im lokalen Speicher erfolgt mit einer AES-CBC-128-Bit-Verschlüsselung.

Unter Android werden die von der EncryptedLocalStorage-Klasse gespeicherten Daten nicht verschlüsselt. Stattdessen werden die Daten durch die vom Betriebssystem bereitgestellte Sicherheit auf Benutzerebene geschützt. Das Android-Betriebssystem weist jeder Anwendung eine separate Benutzer-ID zu. Anwendungen können nur auf ihre eigenen Dateien zugreifen und auf Dateien, die an öffentlichen Speicherorten erstellt wurden (zum Beispiel auf der austauschbaren Speicherkarte). Beachten Sie, dass auf gerooteten Android-Geräten Anwendungen, die mit Root-Berechtigung ausgeführt werden, auf die Dateien anderer Anwendungen zugreifen können. Auf einem gerooteten Gerät bietet der verschlüsselte lokale Speicher deshalb weniger Datensicherheit als auf einem nicht gerooteten Gerät.

Die im verschlüsselten lokalen Speicher enthaltenen Informationen stehen AIR-Anwendungsinhalten nur in der Sicherheits-Sandbox der Anwendung zur Verfügung.

Wenn Sie eine AIR-Anwendung aktualisieren, behält die aktualisierte Version den Zugriff auf alle vorhandenen Daten im verschlüsselten lokalen Speicher, es sei denn:

- Die Elemente wurden hinzugefügt, während ihr `stronglyBound`-Parameter auf `true` gesetzt war
- Die vorhandene und die aktualisierte Version wurden beide vor AIR 1.5.3 veröffentlicht und das Update wurde mit einer Migrationssignatur signiert.

Einschränkungen des verschlüsselten lokalen Speichers

Die Daten im verschlüsselten lokalen Speicher sind durch die Authentifizierungsdaten des Betriebssystems des Benutzers geschützt. Andere Entitäten haben keinen Zugriff auf den Speicher, solange sie sich nicht als der betreffende Benutzer anmelden können. Die Daten sind jedoch nicht vor dem Zugriff durch andere Anwendungen, die von einem authentifizierten Benutzer ausgeführt werden, geschützt.

Da der Benutzer authentifiziert werden muss, damit diese Angriffe funktionieren, sind die vertraulichen Daten eines Benutzers immer noch geschützt (sofern nicht das Benutzerkonto selbst unbefugt verwendet wird). Daten, die Ihre Anwendung vor Benutzern geheim halten möchte, zum Beispiel Lizenzschlüssel oder DRM-Schlüssel, sind jedoch nicht sicher. Deshalb ist der verschlüsselte lokale Speicher nicht der geeignete Ort für die Speicherung dieser Informationen. Er ist lediglich ein geeigneter Speicherort für die persönlichen Daten eines Benutzers, zum Beispiel Kennwörter.

Daten im ELS können aus verschiedenen Gründen verloren gehen. Der Benutzer könnte die Anwendung zum Beispiel deinstallieren und die verschlüsselte Datei löschen. Oder die Herausgeber-ID wird im Rahmen eines Updates geändert. Deshalb sollte der verschlüsselte lokale Speicher als privater Zwischenspeicher eingesetzt werden, nicht jedoch als permanenter Datenspeicher.

Der `stronglyBound`-Parameter ist veraltet und sollte nicht auf `true` gesetzt werden. Das Einstellen des Parameters auf `true` bringt keinerlei zusätzlichen Schutz für Daten. Gleichzeitig geht der Zugriff auf die Daten verloren, wenn die Anwendung aktualisiert wird, selbst wenn die Herausgeber-ID gleich bleibt.

Der verschlüsselte lokale Speicher arbeitet möglicherweise langsamer, wenn mehr als 10 MB Daten gespeichert sind.

Bei der Deinstallation einer AIR-Anwendung werden die Daten im verschlüsselten lokalen Speicher nicht gelöscht.

Empfohlene Verfahren

Zu den bewährten Verfahren bei der Verwendung des verschlüsselten lokalen Speicher gehört Folgendes:

- Speichern Sie vertrauliche Benutzerdaten, wie Kennwörter, im verschlüsselten lokalen Speicher (indem Sie stronglyBound auf false setzen).
- Speichern Sie im verschlüsselten lokalen Speicher keine vertraulichen Anwendungsdaten, wie DRM-Schlüssel oder Lizenz-Token.
- Stellen Sie für Ihre Anwendung eine Möglichkeit bereit, die im ELS gespeicherten Daten wiederherzustellen, falls sie verloren gehen sollten. Dies ist zum Beispiel möglich, indem der Benutzer ggf. zur erneuten Eingabe seiner Benutzerkontodaten aufgefordert wird.
- Verwenden Sie nicht den `stronglyBound`-Parameter.
- Wenn Sie `stronglyBound` auf `true` setzen, migrieren Sie gespeicherte Elemente während eines Updates nicht. Erstellen Sie die Daten stattdessen nach dem Update erneut.
- Speichen Sie nur relativ kleine Datenmengen. Für größere Datenmengen sollten Sie eine AIR SQL-Datenbank mit Verschlüsselung verwenden.

Verwandte Hilfethemen

flash.data.EncryptedLocalStore

Hinzufügen von Daten zum verschlüsselten lokalen Speicher

Speichern Sie Daten unter Verwendung der statischen `setItem()`-Methode der EncryptedLocalStore-Klasse im verschlüsselten lokalen Speicher. Die in einer Hash-Tabelle gespeicherten Daten verwenden Strings als Schlüssel, wobei die Daten als Byte-Arrays gespeichert sind.

Im folgenden Beispielcode wird ein String im verschlüsselten lokalen Speicher gespeichert:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
```

Der dritte Parameter der Methode `setItem()`, der Parameter `stronglyBound`, ist optional. Wenn dieser Parameter auf `true` gesetzt ist, bindet der verschlüsselte lokale Speicher das gespeicherte Element an die digitale Signatur und die Bit der speichernden AIR-Anwendung.

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes, false);
```

Bei einem Element, dessen `stronglyBound`-Parameter beim Speichern auf `true` gesetzt ist, sind nachfolgende Aufrufe von `getItem()` nur erfolgreich, wenn die aufrufende AIR-Anwendung identisch ist mit der speichernden Anwendung (d. h., wenn in den Dateien im Anwendungsverzeichnis keine Daten geändert wurden). Sind die aufrufende und die speichernde Anwendung nicht identisch, wird eine Error-Ausnahme ausgelöst, wenn Sie für ein stark gebundenes Element `getItem()` aufrufen. Wenn Sie die Anwendung aktualisieren, können stark gebundene Daten, die zuvor in den verschlüsselten lokalen Speicher geschrieben wurden, nicht gelesen werden. Das Festlegen von `stronglyBound` auf `true` wird bei mobilen Geräten ignoriert; der Parameter wird immer als `false` behandelt.

Ist der `stronglyBound`-Parameter auf `false` gesetzt (Standardwert), muss nur die Hersteller-ID gleich bleiben, damit die Anwendung die Daten lesen kann. Die Bit der Anwendung können sich ändern (und sie müssen von demselben Herausgeber signiert werden). Es müssen aber nicht dieselben Bit sein wie in der Anwendung, in der die Daten gespeichert wurden. Aktualisierte Anwendungen mit derselben Herausgeber-ID wie das Original können auch weiterhin auf die Daten zugreifen.

Hinweis: *Durch Einstellen von `stronglyBound` auf `true` entsteht in der Praxis kein zusätzlicher Schutz für die Daten. Ein Benutzer mit bösartigen Absichten könnte eine Anwendung nach wie vor ändern, um Zugriff auf die Daten im verschlüsselten lokalen Speicher zu erhalten. Außerdem ist der Schutz der Daten vor externen Bedrohungen, die nicht von Benutzern ausgehen, gleichermaßen wirksam, unabhängig davon, ob `stronglyBound` auf `true` oder `false` eingestellt ist. Deshalb sollte `stronglyBound` nicht auf `true` gesetzt werden.*

Zugriff auf die Daten im verschlüsselten lokalen Speicher

Adobe AIR 1.0 und höher

Sie können mithilfe der Methode `EncryptedLocalStore.getItem()` einen Wert aus dem verschlüsselten lokalen Speicher abrufen. Hier ein Beispiel:

```
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length)); // "Bob"
```

Entfernen von Daten aus dem verschlüsselten lokalen Speicher

Adobe AIR 1.0 und höher

Sie können einen Wert aus dem verschlüsselten lokalen Speicher mithilfe der Methode `EncryptedLocalStore.removeItem()` löschen. Hier ein Beispiel:

```
EncryptedLocalStore.removeItem("firstName");
```

Und durch Aufruf der Methode `EncryptedLocalStore.reset()` können alle Daten aus dem verschlüsselten lokalen Speicher löschen. Auch hierzu ein Beispiel:

```
EncryptedLocalStore.reset();
```

Kapitel 40: Arbeiten mit lokalen SQL-Datenbanken in AIR

Adobe AIR 1.0 und höher

Adobe® AIR® ermöglicht das Erstellen von und Arbeiten mit lokalen SQL-Datenbanken. Die Laufzeitumgebung enthält eine SQL-Datenbank-Engine mit Unterstützung für viele Standard-SQL-Funktionen. Dabei wird das Open-Source-Datenbanksystem SQL Lite verwendet. Eine lokale Datenbank dient der Speicherung lokaler, permanenter Daten. Sie können sie zum Beispiel für Anwendungsdaten, Benutzereinstellungen für die Anwendung, Dokumente oder andere Daten, die die Anwendung lokal speichern soll, verwenden.

Lokale SQL-Datenbanken

Adobe AIR 1.0 und höher

Eine Kurzbeschreibung und Codebeispiele zur Verwendung von SQL-Datenbanken finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Asynchrones Arbeiten mit einer lokalen SQL-Datenbank (Flex)
- Synchrones Arbeiten mit einer lokalen SQL-Datenbank (Flex)
- Verwenden einer verschlüsselten Datenbank (Flex)
- Asynchrones Arbeiten mit einer lokalen SQL-Datenbank (Flash)
- Synchrones Arbeiten mit einer lokalen SQL-Datenbank (Flash)
- Verwenden einer verschlüsselten Datenbank (Flash)

Adobe AIR enthält eine SQL-basierte relationale Datenbank-Engine, die in der Laufzeitumgebung ausgeführt wird. Daten werden lokal in Datenbankdateien auf dem Computer gespeichert, auf dem die AIR-Anwendung ausgeführt wird (zum Beispiel auf der Festplatte des Computers). Da die Datenbank lokal ausgeführt wird und Datendateien lokal gespeichert werden, kann eine Datenbank von der AIR-Anwendung genutzt werden, auch wenn keine Netzwerkverbindung besteht. Somit stellt die lokale SQL-Datenbank-Engine der Laufzeitumgebung eine praktische Möglichkeit zum Speichern permanenter lokaler Anwendungsdaten dar, besonders, wenn Sie bereits Erfahrung im Umgang mit SQL und relationalen Datenbanken haben.

Verwendungszwecke lokaler Datenbanken

Adobe AIR 1.0 und höher

Die lokale SQL-Datenbankfunktion in AIR kann für jeden Zweck eingesetzt werden, für den Sie Anwendungsdaten auf dem lokalen Computer des Benutzers speichern möchten. Adobe AIR beinhaltet mehrere Mechanismen zum lokalen Speichern von Daten, die unterschiedliche Vorteile haben. Nachstehend sind einige Einsatzmöglichkeiten einer lokalen SQL-Datenbank in AIR-Anwendungen aufgeführt:

- Bei einer datenorientierten Anwendung (zum Beispiel ein Adressbuch) kann eine Datenbank zum Speichern der Hauptanwendungsdaten verwendet werden.

- Bei einer dokumentorientierten Anwendung, in der Benutzer Dokumente zum Speichern und ggf. Freigeben erstellen, kann jedes Dokument als eine Datenbankdatei an einem vom Benutzer festgelegten Speicherort gespeichert werden. (Beachten Sie jedoch, dass jede AIR-Anwendung die Datenbankdatei öffnen kann, wenn die Datenbank nicht verschlüsselt ist. Die Verschlüsselung wird für möglicherweise vertrauliche Dokumente empfohlen.)
- Bei einer netzwerkfähigen Anwendung kann eine Datenbank zur Zwischenspeicherung von Anwendungsdaten oder zum temporären Speichern verwendet werden, wenn keine Netzwerkverbindung verfügbar ist. Sie könnten einen Mechanismus zum Synchronisieren der lokalen Datenbanken mit dem Netzwerkdatenspeicher erstellen.
- Bei jeder beliebigen Anwendung lassen sich mit einer Datenbank Anwendungseinstellungen der einzelnen Benutzer, zum Beispiel Benutzeroptionen, oder Anwendungsinformationen wie Fenstergröße und -position speichern.

Verwandte Hilfethemen

Christophe Coenraets: Employee Directory on AIR for Android

Raymond Camden: jQuery and AIR – Moving from web page to application

AIR-Datenbanken und Datenbankdateien

Adobe AIR 1.0 und höher

Eine einzelne lokale Adobe AIR-SQL-Datenbank wird als eine Datei im Dateisystem des Computers gespeichert. Die Laufzeitumgebung enthält die SQL-Datenbank-Engine, die das Erstellen und Strukturieren von Datenbankdateien sowie das Bearbeiten und Abrufen von Dateien aus einer Datenbankdatei verwaltet. Die Laufzeitumgebung legt nicht fest, wie oder wo Datenbankdaten im Dateisystem gespeichert werden; stattdessen wird jede Datenbank vollständig in einer Datei gespeichert. Sie geben den Speicherort im Dateisystem an, an dem die Datenbankdatei gespeichert wird. Eine einzelne AIR-Anwendung kann auf eine oder auf viele separate Datenbanken (d. h. Datenbankdateien) zugreifen. Da die Laufzeitumgebung jede Datenbank als einzelne Datei im Dateisystem speichert, können Sie Ihre Datenbanken so speichern, wie es für Ihre Anwendung und die Dateizugriffsbeschränkungen des Betriebssystems erforderlich ist. Jeder Benutzer kann eine separate Datenbankdatei für seine jeweiligen Daten haben, oder alle Benutzer der Anwendung auf einem Computer greifen auf eine Datenbankdatei mit freigegebenen Daten zu. Da sich die Daten lokal auf einem einzelnen Computer befinden, werden die Daten nicht automatisch für Benutzer auf anderen Computern freigegeben. Die lokale SQL-Datenbank-Engine bietet keine Möglichkeit, SQL-Anweisungen für eine Remote-Datenbank oder serverbasierte Datenbank auszuführen.

Relationale Datenbanken

Adobe AIR 1.0 und höher

Eine relationale Datenbank ist ein Mechanismus zum Speichern (und Abrufen) von Daten auf einem Computer. Daten sind in Tabellen organisiert: Zeilen repräsentieren Datensätze oder Elemente; Spalten (manchmal „Felder“ genannt) teilen jeden Datensatz in einzelne Werte. Eine Adressbuchanwendung könnte zum Beispiel die Tabelle „Freunde“ enthalten. Jede Zeile in der Tabelle steht für einen Freund, der in der Datenbank gespeichert ist. Die Spalten der Tabelle repräsentieren Daten wie Vorname, Nachname, Geburtstag usw. Für jede Zeile mit einem Freund in der Tabelle speichert die Datenbank einen separaten Wert für jede Spalte.

Relationale Datenbanken eigenen sich für die Speicherung komplexer Daten, wobei ein Element mit Elementen eines anderen Typs verknüpft ist – zu diesem in Relation steht. In einer relationalen Datenbank sollten alle Daten, die eine Eins-zu-viele-Beziehung haben (wobei ein einzelner Datensatz mit mehreren Datensätzen eines anderen Typs in Relation steht), auf verschiedene Tabellen verteilt werden. Wenn Sie zum Beispiel in der Adressbuchanwendung mehrere Telefonnummern für jeden Freund speichern möchten, ist dies eine Eins-zu-viele-Beziehung. Die Tabelle „Freunde“ enthält alle persönlichen Informationen für jeden Freund. Eine separate Tabelle „Telefonnummern“ enthält alle Telefonnummern für alle Freunde.

Zusätzlich zur Speicherung der Daten zu Freunden und Telefonnummern benötigt jede Tabelle Daten, um die Beziehung zwischen den beiden Tabellen festzuhalten, damit die Datensätze einzelner Freunde ihren Telefonnummern zugeordnet werden können. Diese Daten werden als ein Primärschlüssel bezeichnet – ein eindeutiger Bezeichner, der jede Zeile in einer Tabelle von anderen Zeilen in dieser Tabelle unterscheidet. Der Primärschlüssel kann ein „natürlicher Schlüssel“ sein, also eines der Datenelemente, die jeden Datensatz in einer Tabelle natürlich auszeichnen. In der Tabelle „Freunde“ können Sie zum Beispiel die Spalte mit dem Geburtsdatum als Primärschlüssel (als natürlichen Schlüssel) verwenden, wenn Sie wissen, dass alle Freunde ein anderes Geburtsdatum haben. Ist kein natürlicher Schlüssel vorhanden, können Sie eine separate Primärschlüsselspalte, zum Beispiel „Freund-ID“, erstellen. Anhand dieses künstlichen Werts kann die Anwendung die einzelnen Zeilen unterscheiden.

Mithilfe eines Primärschlüssels können Sie Beziehungen zwischen verschiedenen Tabellen herstellen. Angenommen, die Tabelle „Freunde“ hat eine Spalte „Freund-ID“, die eine eindeutige Nummer für jede Zeile (für jeden Freund) enthält. Die verwandte Tabelle „Telefonnummern“ kann mit zwei Spalten strukturiert werden; eine mit der „Freund-ID“ des Freundes mit einer bestimmten Telefonnummer, und eine mit der Telefonnummer. Auf diese Weise können alle Freunde, unabhängig von der Anzahl ihrer Telefonnummern, in der Tabelle „Telefonnummern“ gespeichert und mithilfe des Primärschlüssels „Freund-ID“ mit dem entsprechenden Freund verknüpft werden. Wenn ein Primärschlüssel aus einer Tabelle in einer verwandten Tabelle verwendet wird, um die Verbindung zwischen den Datensätzen anzugeben, wird der Wert in der verwandten Tabelle als „Fremdschlüssel“ bezeichnet. Im Gegensatz zu vielen anderen Datenbanken lässt die lokale Datenbank-Engine von AIR es nicht zu, dass Sie Beschränkungen für Fremdschlüssel erstellen. Dies sind Beschränkungen, die automatisch überprüfen, dass ein eingefügter oder aktualisierter Fremdschlüsselwert eine entsprechende Zeile in der Primärschlüsseltabelle hat. Fremdschlüsselbeziehungen sind ein wichtiger Bestandteil der Struktur einer relationalen Datenbank. Fremdschlüssel sollten verwendet werden, wenn Sie Beziehungen zwischen Tabellen in Ihren Datenbank erstellen.

Informationen zu SQL

Adobe AIR 1.0 und höher

SQL (Structured Query Language) wird mit relationalen Datenbanken verwendet, um Daten zu bearbeiten und abzurufen. SQL ist keine prozedurale Sprache, sondern eine beschreibende Sprache. Anstatt dem Computer Anweisungen zu geben, wie die Daten abzurufen sind, beschreibt eine SQL-Anweisung die Daten, die Sie erhalten möchten. Die Datenbank-Engine bestimmt, wie diese Daten abzurufen sind.

SQL wurde vom American National Standards Institute (ANSI) standardisiert. Die lokale SQL-Datenbank von Adobe AIR unterstützt den größten Teil des Standards SQL-92.

Genauere Beschreibungen der in Adobe AIR unterstützten SQL-Sprache finden Sie unter „SQL-Unterstützung in lokalen Datenbanken“ auf Seite 1185.

Informationen zu SQL-Datenbankklassen

Adobe AIR 1.0 und höher

Wenn Sie in ActionScript 3.0 mit lokalen SQL-Datenbanken arbeiten, verwenden Sie Instanzen der folgenden Klassen aus dem flash.data-Paket:

Klasse	Beschreibung
flash.data.SQLConnection	Bietet Möglichkeiten zum Erstellen und Öffnen von Datenbanken (Datenbankdateien) sowie Methoden zum Ausführen von Operationen auf Datenebene und zum Steuern von Datenbanktransaktionen.
flash.data.SQLStatement	Stellt eine einzelne SQL-Anweisung (einen Befehl oder eine Abfrage) dar, die für eine Datenbank ausgeführt wird, und beinhaltet das Definieren des Anweisungstextes und das Festlegen der Parameterwerte.
flash.data.SQLResult	Ermöglicht das Abrufen von Informationen zu oder Ergebnissen aus der Ausführung einer Anweisung, zum Beispiel die Ergebniszeilen einer `SELECT`-Anweisung, die Anzahl der Zeilen, die von einer `UPDATE`- oder `DELETE`-Anweisung betroffen sind.

Um Schemainformationen zu erhalten, die die Struktur einer Datenbank beschreiben, verwenden Sie die folgenden Klassen aus dem flash.data-Paket:

Klasse	Beschreibung
flash.data.SQLSchemaResult	Dient als Container für Datenbankschemaergebnisse, die durch einen Aufruf der `SQLConnection.loadSchema()`-Methode generiert werden.
flash.data.SQLTableSchema	Stellt Informationen bereit, die eine einzelne Tabelle in einer Datenbank beschreiben.
flash.data.SQLViewSchema	Stellt Informationen bereit, die eine einzelne Ansicht in einer Datenbank beschreiben.
flash.data.SQLIndexSchema	Stellt Informationen bereit, die eine einzelne Spalte in einer Tabelle oder Ansicht in einer Datenbank beschreiben.
flash.data.SQLTriggerSchema	Stellt Informationen bereit, die einen einzelnen Auslöser in einer Datenbank beschreiben.

Andere Klassen im flash.data-Paket stellen Konstanten bereit, die mit der SQLConnection-Klasse und der SQLColumnSchema-Klasse verwendet werden:

Klasse	Beschreibung
flash.data.SQLConnection	Definiert eine Gruppe von Konstanten, die die möglichen Werte für den `openMode`-Parameter der Methoden `SQLConnection.open()` und `SQLConnection.openAsync()` darstellen.
flash.data.SQLColumnNameStyle	Definiert eine Gruppe von Konstanten, die die möglichen Werte für die `SQLConnection.columnNameStyle`-Eigenschaft darstellen.
flash.data.SQLTransactionLockType	Definiert eine Gruppe von Konstanten, die die möglichen Werte für den option-Parameter der `SQLConnection.begin()`-Methode darstellen.
flash.data.SQLCollationType	Definiert eine Gruppe von Konstanten, die die möglichen Werte für die `SQLColumnSchema.defaultCollationType`-Eigenschaft und den `defaultCollationType`-Parameter des `SQLColumnSchema()`-Konstruktors darstellen.

Zusätzlich repräsentieren die folgenden Klassen im flash.events-Paket die Ereignisse (und unterstützenden Konstanten), die Sie verwenden:

Klasse	Beschreibung
flash.events.SQLEvent	Definiert die Ereignisse, die eine SQLConnection- oder SQLStatement-Instanz auslöst, wenn eine ihrer Operationen erfolgreich ausgeführt wird. Jeder Operation ist eine Ereignistypkonstante zugewiesen, die in der SQLEvent-Klasse definiert ist.
flash.events.SQLErrorEvent	Definiert das Ereignis, das eine SQLConnection- oder SQLStatement-Instanz auslöst, wenn eine ihrer Operationen zu einem Fehler führt.
flash.events.SQLUpdateEvent	Definiert das Ereignis, das eine SQLConnection-Instanz auslöst, wenn Tabellendaten in einer der verbundenen Datenbanken als Ergebnis einer `INSERT`-, `UPDATE`- oder `DELETE`-Anweisung geändert werden.

Die folgenden Klassen im flash.errors-Paket stellen Informationen zu Fehlern bei Datenbankoperationen bereit:

Klasse	Beschreibung
flash.errors.SQLError	Stellt Informationen zu einem Fehler bei einer Datenbankoperation bereit, darunter die Operation, die ausgeführt werden sollte, sowie die Fehlerursache.
flash.errors.SQLErrorOperation	Definiert eine Gruppe von Konstanten, die die möglichen Werte für die `operation`-Eigenschaft der SQLError-Klasse darstellen. Diese Eigenschaft gibt an, welche Datenbankoperation zu dem Fehler geführt hat.

Synchrone und asynchrone Ausführungsmodi

Adobe AIR 1.0 und höher

Wenn Sie Code für die Arbeit mit einer lokalen SQL-Datenbank schreiben, geben Sie für Datenbankoperationen einen von zwei Ausführungsmodi an: asynchron oder synchron. Im Allgemeinen zeigen die Codebeispiele, wie Sie jeden Vorgang auf beide Arten ausführen, sodass Sie das für Sie am besten geeignete Beispiel verwenden können.

Im asynchronen Modus geben Sie der Laufzeitumgebung eine Anweisung und die Laufzeitumgebung löst ein Ereignis aus, wenn die angeforderte Operation abgeschlossen wurde oder fehlgeschlagen ist. Zunächst weisen Sie die Datenbank-Engine an, eine Operation auszuführen. Die Datenbank-Engine führt die Operation im Hintergrund aus, während die Anwendung weiterhin ausgeführt wird. Wenn die Operation abgeschlossen ist (oder wenn sie fehlschlägt), löst die Datenbank-Engine ein Ereignis aus. Ihr Code, der vom Ereignis ausgelöst wird, führt die nachfolgenden Operationen aus. Dieser Ansatz hat einen wichtigen Vorteil: die Laufzeitumgebung führt die Datenbankoperationen im Hintergrund aus, während der Hauptanwendungscode weiterhin ausgeführt wird. Auch wenn die Datenbankoperation längere Zeit dauert, wird die Anwendung weiter ausgeführt. Der Benutzer kann also mit der Anwendung interagieren, ohne dass der Bildschirm „einfriert“. Der Code für asynchrone Operationen kann komplexer sein als anderer Code. Diese Komplexität ist normalerweise dann gegeben, wenn mehrere voneinander abhängige Operationen zwischen verschiedenen Ereignis-Listener-Methoden aufgeteilt werden müssen.

Vom Konzept her ist es einfacher, Operationen als eine Folge von Schritten, also als eine Reihe synchroner Operationen, zu kodieren als eine Gruppe von Operationen auf verschiedene Ereignis-Listener-Methoden zu verteilen. Neben asynchronen Datenbankoperationen können Sie in Adobe AIR Datenbankoperationen auch synchron ausführen. Im synchronen Ausführungsmodus werden Operationen nicht im Hintergrund ausgeführt. Stattdessen werden sie in derselben Ausführungsabfolge wie der andere Anwendungscode ausgeführt. Sie weisen die Datenbank-Engine an, eine Operation auszuführen. Der Code stoppt dann an diesem Punkt, während die Datenbank-Engine die Operation ausführt. Wenn die Operation abgeschlossen wurde, wird die Ausführung mit der nächsten Codezeile fortgesetzt.

Ob Operationen asynchron oder synchron ausgeführt werden, wird auf der SQLConnection-Ebene festgelegt. Wenn Sie eine einzelne Datenbankverbindung verwenden, können Sie nicht bestimmte Operationen oder Anweisungen synchron ausführen, andere dagegen asynchron. Sie legen fest, ob eine SQLConnection im synchronen oder im asynchronen Ausführungsmodus operiert, indem Sie eine SQLConnection-Methode aufrufen, um die Datenbank zu öffnen. Wenn Sie `SQLConnection.open()` aufrufen, arbeitet die Verbindung im synchronen Ausführungsmodus; wenn Sie `SQLConnection.openAsync()` aufrufen, arbeitet die Verbindung im synchronen Ausführungsmodus. Nachdem eine SQLConnection-Instanz durch Aufruf von `open()` oder `openAsync()` mit einer Datenbank verbunden wurde, ist sie auf den synchronen oder asynchronen Ausführungsmodus festgelegt, solange Sie die Verbindung zur Datenbank nicht schließen und erneut öffnen.

Beide Ausführungsmodi haben ihre Vorteile. In den meisten Punkten ähneln sich die Ausführungsmodi, es gibt jedoch einige Unterschieden, die Sie bei der Arbeit mit den einzelnen Modi beachten sollten. Weitere Informationen zu diesem Themen sowie Vorschläge für die Arbeit in den beiden Modi finden Sie unter „Verwenden von synchronen und asynchronen Datenbankoperationen“ auf Seite 802.

Erstellen und Modifizieren von Datenbanken

Adobe AIR 1.0 und höher

Bevor Ihre Anwendung Daten hinzufügen oder abrufen kann, muss eine Datenbank mit definierten Tabellen vorhanden sein, auf die die Anwendung zugreifen kann. Nachstehend werden die Aufgaben für das Erstellen einer Datenbank und das Erstellen der Datenstruktur in einer Datenbank beschrieben. Auch wenn diese Aufgaben nicht so häufig wie das Einfügen und Abrufen von Daten verwendet werden, sind sie doch für die meisten Anwendungen erforderlich.

Verwandte Hilfethemen

Mind the Flex: Updating an existing AIR database

Erstellen von Datenbanken

Adobe AIR 1.0 und höher

Zur Erstellung einer Datenbankdatei erstellen Sie zunächst eine SQLConnection-Instanz. Sie rufen deren `open()`-Methode auf, um die Verbindung im synchronen Ausführungsmodus zu öffnen, oder die `openAsync()` der Instanz, um den asynchronen Ausführungsmodus zu verwenden. Mit den Methoden `open()` und `openAsync()` wird eine Verbindung zu einer Datenbank hergestellt. Wenn Sie eine File-Instanz übergeben, die für den `reference`-Parameter (dies ist der erste Parameter) auf einen nicht vorhandenen Speicherort verweist, erstellt die `open()`- oder `openAsync()`-Methode eine Datenbankdatei an diesem Speicherort und öffnet eine Verbindung zu der neu erstellten Datenbank.

Unabhängig davon, ob Sie die `open()`-Methode oder die `openAsync()`-Methode zum Erstellen einer Datenbank verwenden, kann der Name der Datenbankdatei ein beliebiger gültiger Dateiname mit einer beliebigen Dateierweiterung sein. Wenn Sie die `open()`- oder `openAsync()`-Methode mit `null` als `reference`-Parameter aufrufen, wird eine In-Memory-Datenbank erstellt statt einer Datenbankdatei auf der Festplatte.

Im nachstehend aufgeführten Code wird der Ablauf beim Erstellen einer Datenbankdatei (einer neuen Datenbank) im asynchronen Ausführungsmodus dargestellt. In diesem Fall wird die Datenbankdatei unter dem Namen „DBSample.db“ gespeichert, siehe „Verweisen auf das Anwendungsspeicherverzeichnis“ auf Seite 716:

```
import flash.data.SQLConnection;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

conn.openAsync(dbFile);

function openHandler(event:SQLEvent):void
{
    trace("the database was created successfully");
}

function errorHandler(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                conn.openAsync(dbFile);
            }

            private function openHandler(event:SQLEvent):void
            {
                trace("the database was created successfully");
            }

            private function errorHandler(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Hinweis: *Mit der File-Klasse können Sie zwar auf einen spezifischen nativen Dateipfad zeigen, damit erhalten Sie möglicherweise jedoch Anwendungen, die nicht auf allen Plattformen funktionieren. Der Pfad C:\Dokumente und Einstellungen\joe\test.db funktioniert zum Beispiel nur unter Windows. Deshalb ist es besser, die statischen Eigenschaften der File-Klasse, wie* `File.applicationStorageDirectory`*, sowie die* `resolvePath()`*-Methode zu verwenden (wie im Beispiel weiter oben). Weitere Informationen finden Sie unter „Pfade von File-Objekten“ auf Seite 712.*

Um Operationen synchron auszuführen, rufen Sie beim Öffnen einer Datenbankverbindung mit der SQLConnection-Instanz die `open()`-Methode auf. Im folgenden Beispiel wird eine SQLConnection-Instanz erstellt und geöffnet, die Operationen synchron ausführt.

```
import flash.data.SQLConnection;
import flash.errors.SQLError;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

try
{
    conn.open(dbFile);
    trace("the database was created successfully");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.errors.SQLError;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                try
                {
                    conn.open(dbFile);
                    trace("the database was created successfully");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Erstellen von Datenbanktabellen

Adobe AIR 1.0 und höher

Zum Erstellen einer Tabelle in einer Datenbank gehört das Ausführen einer SQL-Anweisung für diese Datenbank. Dabei gehen Sie genauso vor wie beim Ausführen einer SQL-Anweisung wie `SELECT`, `INSERT` usw. Um eine Tabelle zu erstellen, verwenden Sie die `CREATE TABLE`-Anweisung, die Definitionen von Spalten und Beschränkungen für die neue Tabelle erhält. Weitere Informationen zum Ausführen von SQL-Anweisungen finden Sie unter „Arbeiten mit SQL-Anweisungen“ auf Seite 777.

Im folgenden Beispiel wird eine Tabelle mit dem Namen „employees“ in einer vorhandenen Datenbankdatei erstellt, wobei der asynchrone Ausführungsmodus verwendet wird. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass eine SQLConnection-Instanz namens `conn` bereits instanziiert wurde und eine Verbindung mit der Datenbank besteht.

```
import flash.data.SQLConnection;
import flash.data.SQLStatement;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;

// ... create and open the SQLConnection instance named conn ...

var createStmt:SQLStatement = new SQLStatement();
createStmt.sqlConnection = conn;

var sql:String =
    "CREATE TABLE IF NOT EXISTS employees (" +
    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
    "   firstName TEXT, " +
    "   lastName TEXT, " +
    "   salary NUMERIC CHECK (salary > 0)" +
    ")";
createStmt.text = sql;

createStmt.addEventListener(SQLEvent.RESULT, createResult);
createStmt.addEventListener(SQLErrorEvent.ERROR, createError);

createStmt.execute();

function createResult(event:SQLEvent):void
{
    trace("Table created");
}

function createError(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLStatement;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                var createStmt:SQLStatement = new SQLStatement();
                createStmt.sqlConnection = conn;

                var sql:String =
                    "CREATE TABLE IF NOT EXISTS employees (" +
                    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "   firstName TEXT, " +
                    "   lastName TEXT, " +
                    "   salary NUMERIC CHECK (salary > 0)" +
                    ")";
                createStmt.text = sql;

                createStmt.addEventListener(SQLEvent.RESULT, createResult);
                createStmt.addEventListener(SQLErrorEvent.ERROR, createError);

                createStmt.execute();
            }

            private function createResult(event:SQLEvent):void
            {
                trace("Table created");
            }

            private function createError(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Im folgenden Beispiel wird eine Tabelle mit dem Namen „employees“ in einer vorhandenen Datenbankdatei erstellt, wobei der synchrone Ausführungsmodus verwendet wird. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass eine SQLConnection-Instanz namens `conn` bereits instanziiert wurde und eine Verbindung mit der Datenbank besteht.

```
import flash.data.SQLConnection;
import flash.data.SQLStatement;
import flash.errors.SQLError;

// ... create and open the SQLConnection instance named conn ...

var createStmt:SQLStatement = new SQLStatement();
createStmt.sqlConnection = conn;

var sql:String =
    "CREATE TABLE IF NOT EXISTS employees (" +
    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
    "   firstName TEXT, " +
    "   lastName TEXT, " +
    "   salary NUMERIC CHECK (salary > 0)" +
    ")";
createStmt.text = sql;

try
{
    createStmt.execute();
    trace("Table created");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                var createStmt:SQLStatement = new SQLStatement();
                createStmt.sqlConnection = conn;

                var sql:String =
                    "CREATE TABLE IF NOT EXISTS employees (" +
                    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "   firstName TEXT, " +
                    "   lastName TEXT, " +
                    "   salary NUMERIC CHECK (salary > 0)" +
                    ")";
                createStmt.text = sql;

                try
                {
                    createStmt.execute();
                    trace("Table created");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Bearbeiten von SQL-Datenbankdaten

Adobe AIR 1.0 und höher

Es gibt einige allgemeine Aufgaben, die Sie ausführen, wenn Sie mit lokalen Datenbanken arbeiten. Zu diesen Aufgaben gehört das Herstellen einer Datenbankverbindung, das Hinzufügen von Daten zu Tabellen in einer Datenbank und das Abrufen von Daten aus Tabellen. Bei der Ausführung dieser Aufgaben sollten Sie verschiedene Punkte beachten, zum Beispiel das Arbeiten mit Datentypen und den Umgang mit Fehlern.

Beachten Sie, dass es verschiedene Datenbankaufgaben gibt, mit denen Sie sich selten befassen müssen, die aber manchmal auszuführen sind, bevor Sie häufigere Aufgaben ausführen können. Zum Beispiel müssen Sie die Datenbank und die Tabellenstruktur in der Datenbank erstellen, bevor Sie die Verbindung zu einer Datenbank herstellen und Daten aus einer Tabelle abrufen können. Diese ersten Einrichtungsaufgaben werden unter „Erstellen und Modifizieren von Datenbanken“ auf Seite 766 beschrieben.

Sie können festlegen, dass Datenbankoperationen asynchron ausgeführt werden. In diesem Fall wird die Datenbank-Engine im Hintergrund ausgeführt und benachrichtigt Sie durch das Auslösen eines Ereignisses, wenn die Operation erfolgreich abgeschlossen wurde oder fehlgeschlagen ist. Sie können diese Operationen auch synchron ausführen. Dabei werden die Datenbankoperationen nacheinander ausgeführt und die gesamte Anwendung (einschließlich Bildschirmaktualisierungen) wartet, bis die Operationen abgeschlossen sind, bevor weiterer Code ausgeführt wird. Weitere Informationen zum Arbeiten im asynchronen oder synchronen Ausführungsmodus finden Sie unter „Verwenden von synchronen und asynchronen Datenbankoperationen“ auf Seite 802.

Herstellen von Verbindungen mit Datenbanken

Adobe AIR 1.0 und höher

Bevor Sie Datenbankoperationen ausführen können, müssen Sie zunächst eine Verbindung zur Datenbankdatei herstellen. Eine SQLConnection-Instanz repräsentiert eine Verbindung mit einer oder mehreren Datenbanken. Die erste Datenbank, die mithilfe einer SQLConnection-Instanz verbunden wird, wird als Hauptdatenbank bezeichnet. Zu dieser Datenbank wird die Verbindung mit der `open()`-Methode (für synchronen Ausführungsmodus) oder mit der `openAsync()`-Methode (für asynchronen Ausführungsmodus) hergestellt.

Wenn Sie eine Datenbank mit `openAsync()` öffnen, registrieren Sie einen Listener für das `open`-Ereignis der SQLConnection-Instanz, damit Sie benachrichtigt werden, wenn die `openAsync()`-Operation abgeschlossen ist. Registrieren Sie einen Listener für das `error`-Ereignis der SQLConnection-Instanz, um festzustellen, wenn die Operation fehlschlägt.

Im folgenden Beispiel wird eine vorhandene Datenbankdatei für die asynchrone Ausführung geöffnet. Die Datenbankdatei heißt „DBSample.db“ und befindet sich im Anwendungsspeicherverzeichnis des Benutzers, siehe „Verweisen auf das Anwendungsspeicherverzeichnis“ auf Seite 716.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

conn.openAsync(dbFile, SQLMode.UPDATE);

function openHandler(event:SQLEvent):void
{
    trace("the database opened successfully");
}

function errorHandler(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLMode;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                conn.openAsync(dbFile, SQLMode.UPDATE);
            }

            private function openHandler(event:SQLEvent):void
            {
                trace("the database opened successfully");
            }

            private function errorHandler(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Im folgenden Beispiel wird eine vorhandene Datenbankdatei für die synchrone Ausführung geöffnet. Die Datenbankdatei heißt „DBSample.db“ und befindet sich im Anwendungsspeicherverzeichnis des Benutzers, siehe „Verweisen auf das Anwendungsspeicherverzeichnis“ auf Seite 716.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.errors.SQLError;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

try
{
    conn.open(dbFile, SQLMode.UPDATE);
    trace("the database opened successfully");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLMode;
            import flash.errors.SQLError;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                try
                {
                    conn.open(dbFile, SQLMode.UPDATE);
                    trace("the database opened successfully");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Beachten Sie, dass beim Aufruf der `openAsync()`-Methode im Beispiel für die asynchrone Ausführung und beim Aufruf der `open()`-Methode im Beispiel für die synchrone Ausführung das zweite Argument die Konstante `SQLMode.UPDATE` ist. Wenn Sie `SQLMode.UPDATE` für den zweiten Parameter (`openMode`) angeben, gibt die Laufzeitumgebung einen Fehler aus, wenn die angegebene Datei nicht vorhanden ist. Wenn Sie `SQLMode.CREATE` für den `openMode` übergeben (oder wenn Sie den `openMode`-Parameter auslassen), versucht die Laufzeitumgebung eine Datenbankdatei zu erstellen, wenn die angegebene Datei nicht vorhanden ist. Wenn die Datei vorhanden ist, wird sie jedoch geöffnet, was der Verwendung von `SQLMode.Update` entspricht. Sie können auch `SQLMode.READ` für den `openMode`-Parameter angeben, um eine vorhandene Datenbank im schreibgeschützten Modus zu öffnen. In diesem Fall können Sie zwar Daten aus der Datenbank abrufen, aber keine Daten hinzufügen, löschen oder ändern.

Arbeiten mit SQL-Anweisungen

Adobe AIR 1.0 und höher

Eine einzelne SQL-Anweisung (Abfrage oder Befehl) wird in der Laufzeitumgebung als SQLStatement-Objekt dargestellt. Gehen Sie wie nachstehend beschrieben vor, um eine SQL-Anweisung zu erstellen und auszuführen:

Erstellen Sie eine SQLStatement-Instanz.
Das SQLStatement-Objekt repräsentiert die SQL-Anweisung in Ihrer Anwendung.

```
var selectData:SQLStatement = new SQLStatement();
```

Geben Sie an, mit welcher Datenbank die Abfrage ausgeführt werden soll.
Setzen Sie dazu die `sqlConnection`-Eigenschaft des SQLStatement-Objekts auf die SQLConnection-Instanz, die mit der gewünschten Datenbank verbunden ist.

```
// A SQLConnection named "conn" has been created previously
selectData.sqlConnection = conn;
```

Geben Sie die SQL-Anweisung an.
Erstellen Sie den Text der Anweisung als String und weisen Sie ihn der `text`-Eigenschaft der SQLStatement-Instanz zu.

```
selectData.text = "SELECT col1, col2 FROM my_table WHERE col1 = :param1";
```

Definieren Sie Funktionen, um das Ergebnis der Ausführungsoperation zu verarbeiten (nur für den asynchronen Ausführungsmodus).
Registrieren Sie mit der `addEventListener()`-Methode Funktionen als Listener für die Ereignisse `result` und `error` der SQLStatement-Instanz.

```
// using listener methods and addEventListener()

selectData.addEventListener(SQLEvent.RESULT, resultHandler);
selectData.addEventListener(SQLErrorEvent.ERROR, errorHandler);

function resultHandler(event:SQLEvent):void
{
    // do something after the statement execution succeeds
}

function errorHandler(event:SQLErrorEvent):void
{
    // do something after the statement execution fails
}
```

Alternativ dazu können Sie Listener-Methoden mithilfe eines Responder-Objekts angeben. In diesem Fall erstellen Sie die Responder-Instanz und verknüpfen die Listener-Methoden damit.

```
// using a Responder (flash.net.Responder)

var selectResponder = new Responder(onResult, onError);

function onResult(result:SQLResult):void
{
    // do something after the statement execution succeeds
}

function onError(error:SQLError):void
{
    // do something after the statement execution fails
}
```

Wenn der Text der Anweisung Parameter enthält, weisen Sie diesen Parametern Werte zu.
Um Parameterwerte zuzuweisen, verwenden Sie die assoziative Array-Eigenschaft `parameters` der SQLStatement-Instanz.

```
selectData.parameters[":param1"] = 25;
```

Führen Sie die SQL-Anweisung aus.
Rufen Sie die `execute()`-Methode der SQLStatement-Instanz auf.

```
// using synchronous execution mode
// or listener methods in asynchronous execution mode
selectData.execute();
```

Wenn Sie im asynchronen Modus anstelle von Ereignis-Listenern einen Responder verwenden, übergeben Sie der `execute()`-Methode zusätzlich die Responder-Instanz.

```
// using a Responder in asynchronous execution mode
selectData.execute(-1, selectResponder);
```

Beispiele zur Veranschaulichung dieser Schritte finden Sie unter den folgenden Themen:

„Abrufen von Daten aus einer Datenbank“ auf Seite 781

„Einfügen von Daten“ auf Seite 791

„Ändern oder Löschen von Daten“ auf Seite 797

Verwenden von Parametern in Anweisungen

Adobe AIR 1.0 und höher

Mit SQL-Anweisungsparametern können Sie eine wiederverwendbare SQL-Anweisung erstellen. Wenn Sie Anweisungsparameter verwenden, können sich die Werte in der Anweisung ändern (zum Beispiel Werte, die in einer `INSERT`-Anweisung hinzugefügt werden), der grundlegende Text der Anweisung bleibt jedoch unverändert. Die Verwendung von Parametern bietet also Leistungsvorteile und vereinfacht das Kodieren von Anwendungen.

Funktionsweise von Anweisungsparametern

Adobe AIR 1.0 und höher

Häufig wird eine einzelne SQL-Anweisung mehrmals in einer Anwendung verwendet, wobei sie jedes Mal leicht verändert wird. Stellen Sie sich zum Beispiel eine Anwendung zur Inventarisierung vor, in der ein Benutzer neue Bestände in die Datenbank eingeben kann. Der Anwendungscode, mit dem ein Bestandselement zur Datenbank hinzugefügt wird, führt eine `INSERT`-SQL-Anweisung aus, mit der Daten zur Datenbank hinzugefügt werden. Jedes Mal, wenn die Anweisung ausgeführt wird, gibt es jedoch eine kleine Änderung. Die eigentlichen Werte, die in die Tabelle eingefügt werden, unterscheiden sich, da sie spezifisch für das hinzugefügte Bestandselement sind.

Wenn Sie eine SQL-Anweisung mehrere Male mit jeweils unterschiedlichen Werten in der Anweisung verwenden, schreiben Sie am besten eine SQL-Anweisung, die Parameter anstelle von Literalwerten im SQL-Text enthält. Ein Parameter ist ein Platzhalter im Text der Anweisung, der bei jeder Ausführung der Anweisung durch einen tatsächlichen Wert ersetzt wird. Zur Verwendung von Parametern in einer SQL-Anweisung erstellen Sie die SQLStatement-Instanz wie gewohnt. Für die SQL-Anweisung, die der `text`-Eigenschaft zugewiesen ist, verwenden Sie Parameterplatzhalter anstelle von Literalwerten. Sie definieren dann den Wert für jeden Parameter, indem Sie den Wert eines Elements für die `parameters`-Eigenschaft der SQLStatement-Instanz einstellen. Die `parameters`-Eigenschaft ist ein assoziatives Array, deshalb legen Sie einen bestimmten Wert mithilfe der folgenden Syntax fest:

```
statement.parameters[parameter_identifier] = value;
```

Der *parameter_identifier* ist ein String, wenn Sie einen benannten Parameter verwenden, oder eine Ganzzahlenindexnummer, wenn Sie einen unbenannten Parameter verwenden.

Verwenden benannter Parameter

Adobe AIR 1.0 und höher

Ein Parameter kann ein benannter Parameter sein. Ein benannter Parameter hat einen bestimmten Namen, den die Datenbank verwendet, um den Parameterwert seiner Platzhalterposition im Text der Anweisung zuzuweisen. Ein Parametername besteht aus dem Zeichen „:“ oder „@“ gefolgt von einem Namen, wie in den folgenden Beispielen:

```
:itemName
@firstName
```

Im folgenden Code wird die Verwendung benannter Parameter veranschaulicht:

```
var sql:String =
    "INSERT INTO inventoryItems (name, productCode)" +
    "VALUES (:name, :productCode)";

var addItemStmt:SQLStatement = new SQLStatement();
addItemStmt.sqlConnection = conn;
addItemStmt.text = sql;

// set parameter values
addItemStmt.parameters[":name"] = "Item name";
addItemStmt.parameters[":productCode"] = "12345";

addItemStmt.execute();
```

Verwenden unbenannter Parameter

Adobe AIR 1.0 und höher

Alternativ zu benannten Parametern können Sie auch unbenannte Parameter verwenden. Um einen unbenannten Parameter zu verwenden, kennzeichnen Sie einen Parameter in einer SQL-Anweisung mit einem „?“ (Fragezeichen). Jedem Parameter wird entsprechend der Reihenfolge der Parameter in der Anweisung ein numerischer Index zugewiesen. Begonnen wird mit dem Index 0 für den ersten Parameter. Das folgende Beispiel ist eine Variante des vorstehenden Beispiels. In dieser Version werden unbenannte Parameter verwendet:

```
var sql:String =
    "INSERT INTO inventoryItems (name, productCode)" +
    "VALUES (?, ?)";

var addItemStmt:SQLStatement = new SQLStatement();
addItemStmt.sqlConnection = conn;
addItemStmt.text = sql;

// set parameter values
addItemStmt.parameters[0] = "Item name";
addItemStmt.parameters[1] = "12345";

addItemStmt.execute();
```

Vorteile bei der Verwendung von Parametern

Adobe AIR 1.0 und höher

Die Verwendung von Parametern in einer SQL-Anweisung hat mehrere Vorteile:

Bessere Leistung Eine SQLStatement-Instanz, die Parameter enthält, ist effizienter im Vergleich mit einer Anweisung, bei der der SQL-Text bei jeder Ausführung dynamisch erstellt wird. Die Leistung wird verbessert, da die Anweisung nur ein einziges Mal vorbereitet werden muss und dann viele Male mit unterschiedlichen Parameterwerten ausgeführt werden kann, ohne dass die SQL-Anweisung neu kompiliert werden muss.

Explizite Datentypisierung Parameter dienen zur typisierten Ersetzung von Werten, die bei der Konstruktion der SQL-Anweisung unbekannt sind. Nur mithilfe von Parametern kann die Speicherklasse eines an die Datenbank übergebenen Werts sichergestellt werden. Ohne Parameter versucht die Laufzeitumgebung, alle Werte auf der Basis der Typenaffinität der zugewiesenen Spalte von ihrer Textrepräsentation in eine Speicherklasse zu konvertieren.

Weitere Informationen zu Speicherklassen und zur Spaltenaffinität finden Sie unter „Unterstützte Datentypen“ auf Seite 1209.

Mehr Sicherheit Die Verwendung von Parametern trägt dazu bei, böswillige Angriffe zu vermeiden, die als SQL-Injection (SQL-Einschleusung) bezeichnet werden. Bei einem SQL-Injection-Angriff gibt ein Benutzer einen SQL-Code an einer für Benutzer zugänglichen Stelle ein (z. B. ein Dateneingabefeld). Wenn der Anwendungscode eine SQL-Anweisung durch die direkte Verkettung der Benutzereingabe mit dem SQL-Text erstellt, wird der vom Benutzer eingegebene SQL-Code an der Datenbank ausgeführt. Das folgende Beispiel zeigt die Verkettung der Benutzereingabe mit dem SQL-Text. **Verwenden Sie dieses Verfahren nicht**:

```
// assume the variables "username" and "password"
// contain user-entered data

var sql:String =
    "SELECT userId " +
    "FROM users " +
    "WHERE username = '" + username + "' " +
    "    AND password = '" + password + "'";

var statement:SQLStatement = new SQLStatement();
statement.text = sql;
```

Indem Sie Anweisungsparameter verwenden, anstatt vom Benutzer eingegebene Werte zum Text einer Anweisung zu verketten, verhindern Sie SQL-Injection-Angriffe. Die SQL-Injection ist in diesem Fall nicht möglich, da die Parameterwerte explizit als Ersatzwerte behandelt werden, anstatt Teil des literalen Anweisungstexts zu werden. Folgendes ist die empfohlene Alternative zur vorherigen Notierung:

```
// assume the variables "username" and "password"
// contain user-entered data

var sql:String =
    "SELECT userId " +
    "FROM users " +
    "WHERE username = :username " +
    "    AND password = :password";

var statement:SQLStatement = new SQLStatement();
statement.text = sql;

// set parameter values
statement.parameters[":username"] = username;
statement.parameters[":password"] = password;
```

Abrufen von Daten aus einer Datenbank

Adobe AIR 1.0 und höher

Das Abrufen von Daten aus einer Datenbank beinhaltet zwei Schritte. Zunächst führen Sie eine SQL-`SELECT`-Anweisung aus, mit der Sie die Gruppe von Daten beschreiben, die Sie aus Datenbank abrufen möchten. Als Nächstes greifen Sie auf die abgerufenen Daten zu und zeigen sie an oder bearbeiten sie, wie für Ihre Anwendung erforderlich.

Ausführen einer SELECT-Anweisung

Adobe AIR 1.0 und höher

Zum Abrufen vorhandener Daten aus einer Datenbank verwenden Sie eine SQLStatement-Instanz. Weisen Sie der `text`-Eigenschaft der Instanz die entsprechende SQL-`SELECT`-Anweisung zu und rufen Sie dann ihre `execute()`-Methode auf.

Einzelheiten zur Syntax der `SELECT`-Anweisung finden Sie unter „SQL-Unterstützung in lokalen Datenbanken“ auf Seite 1185.

Im folgenden Beispiel wird eine `SELECT`-Anweisung ausgeführt, um Daten aus der Tabelle „products“ abzurufen, wobei der asynchrone Ausführungsmodus verwendet wird:

```
var selectStmt:SQLStatement = new SQLStatement();

// A SQLConnection named "conn" has been created previously
selectStmt.sqlConnection = conn;

selectStmt.text = "SELECT itemId, itemName, price FROM products";

selectStmt.addEventListener(SQLEvent.RESULT, resultHandler);
selectStmt.addEventListener(SQLErrorEvent.ERROR, errorHandler);

selectStmt.execute();

function resultHandler(event:SQLEvent):void
{
    var result:SQLResult = selectStmt.getResult();

    var numResults:int = result.data.length;
    for (var i:int = 0; i < numResults; i++)
    {
        var row:Object = result.data[i];
        var output:String = "itemId: " + row.itemId;
        output += "; itemName: " + row.itemName;
        output += "; price: " + row.price;
        trace(output);
    }
}

function errorHandler(event:SQLErrorEvent):void
{
    // Information about the error is available in the
    // event.error property, which is an instance of
    // the SQLError class.
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                var selectStmt:SQLStatement = new SQLStatement();

                // A SQLConnection named "conn" has been created previously
                selectStmt.sqlConnection = conn;

                selectStmt.text = "SELECT itemId, itemName, price FROM products";

                selectStmt.addEventListener(SQLEvent.RESULT, resultHandler);
                selectStmt.addEventListener(SQLErrorEvent.ERROR, errorHandler);
```

```
            selectStmt.execute();
        }

        private function resultHandler(event:SQLEvent):void
        {
            var result:SQLResult = selectStmt.getResult();

            var numResults:int = result.data.length;
            for (var i:int = 0; i < numResults; i++)
            {
                var row:Object = result.data[i];
                var output:String = "itemId: " + row.itemId;
                output += "; itemName: " + row.itemName;
                output += "; price: " + row.price;
                trace(output);
            }
        }

        private function errorHandler(event:SQLErrorEvent):void
        {
            // Information about the error is available in the
            // event.error property, which is an instance of
            // the SQLError class.
        }
    ]]>
  </mx:Script>
</mx:WindowedApplication>
```

Im folgenden Beispiel wird eine `SELECT`-Anweisung ausgeführt, um Daten aus der Tabelle „products“ abzurufen, wobei der synchrone Ausführungsmodus verwendet wird:

```
var selectStmt:SQLStatement = new SQLStatement();

// A SQLConnection named "conn" has been created previously
selectStmt.sqlConnection = conn;

selectStmt.text = "SELECT itemId, itemName, price FROM products";

try
{
    selectStmt.execute();

    var result:SQLResult = selectStmt.getResult();

    var numResults:int = result.data.length;
    for (var i:int = 0; i < numResults; i++)
    {
        var row:Object = result.data[i];
        var output:String = "itemId: " + row.itemId;
        output += "; itemName: " + row.itemName;
        output += "; price: " + row.price;
        trace(output);
    }
}
catch (error:SQLError)
{
    // Information about the error is available in the
    // error variable, which is an instance of
    // the SQLError class.
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                var selectStmt:SQLStatement = new SQLStatement();

                // A SQLConnection named "conn" has been created previously
                selectStmt.sqlConnection = conn;

                selectStmt.text = "SELECT itemId, itemName, price FROM products";

                try
                {
                    selectStmt.execute();

                    var result:SQLResult = selectStmt.getResult();
```

```
                var numResults:int = result.data.length;
                for (var i:int = 0; i < numResults; i++)
                {
                    var row:Object = result.data[i];
                    var output:String = "itemId: " + row.itemId;
                    output += "; itemName: " + row.itemName;
                    output += "; price: " + row.price;
                    trace(output);
                }
            }
            catch (error:SQLError)
            {
                // Information about the error is available in the
                // error variable, which is an instance of
                // the SQLError class.
            }
        }
    ]]>
</mx:Script>
</mx:WindowedApplication>
```

Im asynchronen Modus löst die SQLStatement-Instanz nach dem Ausführen der Anweisung ein `result`-Ereignis aus (`SQLEvent.RESULT`) und zeigt damit an, dass die Anweisung erfolgreich ausgeführt wurde. Alternativ dazu, falls ein Responder-Objekt als Argument an die `execute()`-Methode übergeben wird, wird die Ergebnisprozedurfunktion des Responder-Objekts aufgerufen. Im synchronen Ausführungsmodus wird die Ausführung angehalten, bis die `execute()`-Operation abgeschlossen ist, erst danach wird die nächste Codezeile ausgeführt.

Zugriff auf die Ergebnisdaten der SELECT-Anweisung

Adobe AIR 1.0 und höher

Nachdem die Ausführung der `SELECT`-Anweisung beendet wurde, wird im nächsten Schritt auf die abgerufenen Daten zugegriffen. Sie rufen die Ergebnisdaten nach dem Ausführen einer `SELECT`-Anweisung ab, indem Sie die `getResult()`-Methode des SQLStatement-Objekts aufrufen:

```
var result:SQLResult = selectStatement.getResult();
```

Die `getResult()`-Methode gibt ein SQLResult-Objekt zurück. Die `data`-Eigenschaft des SQLResult-Objekts ist ein Array, das die Ergebnisse der `SELECT`-Anweisung enthält:

```
var numResults:int = result.data.length;
for (var i:int = 0; i < numResults; i++)
{
    // row is an Object representing one row of result data
    var row:Object = result.data[i];
}
```

Jede Datenzeile im `SELECT`-Ergebnissatz wird eine Object-Instanz im `data`-Array. Dieses Objekt verfügt über Eigenschaften, deren Namen mit den Spaltennamen des Ergebnissatzes übereinstimmen. Die Eigenschaften enthalten die Werte aus den Spalten des Ergebnissatzes. Angenommen, eine `SELECT`-Anweisung ergibt einen Ergebnissatz mit drei Spalten namens „itemId“ (Artikel-ID), „itemName“ (Artikelname) und „price“ (Preis). Für jede Zeile im Ergebnissatz wird eine Object-Instanz erstellt, die über die Eigenschaften `itemId`, `itemName` und `price` verfügt. Diese Eigenschaften enthalten die Werte aus den entsprechenden Spalten.

Im folgenden Codebeispiel wird eine SQLStatement-Instanz definiert, deren Text eine `SELECT`-Anweisung ist. Die Anweisung ruft Zeilen mit den Werten der Spalten `firstName` und `lastName` aus allen Zeilen einer Tabelle mit dem Namen `employees` ab. In diesem Beispiel wird der asynchrone Ausführungsmodus verwendet. Wenn die Ausführung abgeschlossen ist, wird die `selectResult()`-Methode aufgerufen und der Zugriff auf die resultierenden Datenzeilen erfolgt über `SQLStatement.getResult()`. Mit der `trace()`-Methode werden die Daten angezeigt. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass eine SQLConnection-Instanz mit dem Namen `conn` bereits instanziiert wurde und eine Verbindung mit der Datenbank besteht. Des Weiteren wird vorausgesetzt, dass die Tabelle „employees" bereits erstellt und mit Daten gefüllt wurde.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var selectStmt:SQLStatement = new SQLStatement();
selectStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "SELECT firstName, lastName " +
    "FROM employees";
selectStmt.text = sql;

// register listeners for the result and error events
selectStmt.addEventListener(SQLEvent.RESULT, selectResult);
selectStmt.addEventListener(SQLErrorEvent.ERROR, selectError);

// execute the statement
selectStmt.execute();

function selectResult(event:SQLEvent):void
{
    // access the result data
    var result:SQLResult = selectStmt.getResult();

    var numRows:int = result.data.length;
    for (var i:int = 0; i < numRows; i++)
    {
        var output:String = "";
        for (var columnName:String in result.data[i])
        {
            output += columnName + ": " + result.data[i][columnName] + "; ";
        }
        trace("row[" + i.toString() + "]\t", output);
    }
}

function selectError(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var selectStmt:SQLStatement = new SQLStatement();
                selectStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "SELECT firstName, lastName " +
                    "FROM employees";
                selectStmt.text = sql;

                // register listeners for the result and error events
                selectStmt.addEventListener(SQLEvent.RESULT, selectResult);
                selectStmt.addEventListener(SQLErrorEvent.ERROR, selectError);

                // execute the statement
                selectStmt.execute();
            }

            private function selectResult(event:SQLEvent):void
            {
                // access the result data
                var result:SQLResult = selectStmt.getResult();

                var numRows:int = result.data.length;
                for (var i:int = 0; i < numRows; i++)
                {
                    var output:String = "";
                    for (var columnName:String in result.data[i])
                    {
                        output += columnName + ": " + result.data[i][columnName] + "; ";
                    }
                    trace("row[" + i.toString() + "]\t", output);
                }
            }

            private function selectError(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Im folgenden Beispiel wird dieselbe Vorgehensweise wie im vorstehenden Beispiel veranschaulicht, allerdings wird hier der synchrone Ausführungsmodus verwendet. Im Beispiel wird eine SQLStatement-Instanz definiert, deren Text eine `SELECT`-Anweisung ist. Die Anweisung ruft Zeilen mit den Werten der Spalten `firstName` und `lastName` aus allen Zeilen einer Tabelle mit dem Namen `employees` ab. Auf die resultierenden Datenzeilen wird mit `SQLStatement.getResult()` zugegriffen; die Anzeige erfolgt über die `trace()`-Methode. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass eine SQLConnection-Instanz mit dem Namen `conn` bereits instanziiert wurde und eine Verbindung mit der Datenbank besteht. Des Weiteren wird vorausgesetzt, dass die Tabelle „employees“ bereits erstellt und mit Daten gefüllt wurde.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.errors.SQLError;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var selectStmt:SQLStatement = new SQLStatement();
selectStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "SELECT firstName, lastName " +
    "FROM employees";
selectStmt.text = sql;

try
{
    // execute the statement
    selectStmt.execute();

    // access the result data
    var result:SQLResult = selectStmt.getResult();

    var numRows:int = result.data.length;
    for (var i:int = 0; i < numRows; i++)
    {
        var output:String = "";
        for (var columnName:String in result.data[i])
        {
            output += columnName + ": " + result.data[i][columnName] + "; ";
        }
        trace("row[" + i.toString() + "]\t", output);
    }
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var selectStmt:SQLStatement = new SQLStatement();
                selectStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "SELECT firstName, lastName " +
                    "FROM employees";
                selectStmt.text = sql;

                try
                {
                    // execute the statement
                    selectStmt.execute();

                    // access the result data
                    var result:SQLResult = selectStmt.getResult();

                    var numRows:int = result.data.length;
                    for (var i:int = 0; i < numRows; i++)
                    {
                        var output:String = "";
                        for (var columnName:String in result.data[i])
                        {
                            output += columnName + ": ";
                            output += result.data[i][columnName] + "; ";
                        }
                        trace("row[" + i.toString() + "]\t", output);
                    }
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Definieren des Datentyps von SELECT-Ergebnisdaten

Adobe AIR 1.0 und höher

Standardmäßig wird jede Zeile, die von einer `SELECT`-Anweisung zurückgegeben wird, als eine Objektinstanz mit Eigenschaften, die nach den Spalten des Ergebnissatzes benannt sind, und mit dem Wert der einzelnen Spalten als Wert der jeweiligen Eigenschaft erstellt. Bevor Sie eine SQL-`SELECT`-Anweisung ausführen, können Sie jedoch die `itemClass`-Eigenschaft der SQLStatement-Instanz auf eine Klasse setzen. Indem Sie die `itemClass`-Eigenschaft festlegen, wird jede von der `SELECT`-Anweisung zurückgegebene Zeile als eine Instanz der jeweiligen Klasse erstellt. Die Laufzeitumgebung weist den Eigenschaften die Werte der Ergebnisspalten zu, indem die Spaltennamen im `SELECT`-Ergebnis auf die Namen der Eigenschaften in der `itemClass`-Klasse gesetzt werden.

Jede Klasse, die als ein `itemClass`-Eigenschaftenwert zugewiesen wird, muss über einen Konstruktor verfügen, der keine Parameter benötigt. Außerdem muss die Klasse je eine Eigenschaft für jede von der `SELECT`-Anweisung zurückgegebene Spalte aufweisen. Es gilt als Fehler, wenn es für eine Spalte der Liste `SELECT` keinen entsprechenden Eigenschaftsnamen in der Klasse `itemClass` gibt.

Abrufen von SELECT-Ergebnissen in Teilen

Adobe AIR 1.0 und höher

Standardmäßig ruft eine `SELECT`-Anweisung alle Zeilen des Ergebnissatzes gleichzeitig ab. Nachdem die Anweisung abgeschlossen ist, verarbeiten Sie die Daten normalerweise auf die eine oder andere Art; Sie erstellen zum Beispiel Objekte oder zeigen die Daten auf dem Bildschirm an. Wenn die Anweisung eine große Anzahl von Zeilen zurückgibt, kann die gleichzeitige Verarbeitung aller Daten sehr rechenintensiv sein, sodass die Benutzeroberfläche unter Umständen nicht aktualisiert wird.

Sie können die wahrgenommene Leistung Ihrer Anwendung verbessern, indem Sie die Laufzeitumgebung anweisen, jeweils nur eine bestimmte Anzahl von Ergebniszeilen zurückzugeben. Auf diese Weise werden die ersten Ergebnisdaten schneller zurückgegeben. Außerdem können Sie die Ergebniszeilen in Gruppen unterteilen, sodass die Benutzeroberfläche aktualisiert wird, nachdem die einzelnen Gruppen von Zeilen verarbeitet wurden. Beachten Sie, dass sich dieses Verfahren nur für den asynchronen Ausführungsmodus eignet.

Um `SELECT`-Ergebnisse in Teilen abzurufen, geben Sie einen Wert für den ersten Parameter (den `prefetch`-Parameter) der `SQLStatement.execute()`-Methode an. Der `prefetch`-Parameter gibt die Anzahl der Zeilen an, die beim ersten Ausführen der Anweisung abgerufen werden sollen. Wenn Sie für eine SQLStatement-Instanz die `execute()`-Methode aufrufen, geben Sie einen Wert für den `prefetch`-Parameter an, um nur diese Menge an Zeilen abzurufen:

```
var stmt:SQLStatement = new SQLStatement();
stmt.sqlConnection = conn;

stmt.text = "SELECT ...";

stmt.addEventListener(SQLEvent.RESULT, selectResult);

stmt.execute(20); // only the first 20 rows (or fewer) are returned
```

Die Anweisung löst das `result`-Ereignis aus und zeigt damit an, dass die erste Gruppe von Ergebniszeilen verfügbar ist. Die resultierende SQLResult-Instanz hat die `data`-Eigenschaft, die die Datenzeilen enthält. Die `complete`-Eigenschaft gibt an, ob noch weitere abzurufende Ergebniszeilen vorhanden sind. Um weitere Ergebniszeilen abzurufen, rufen Sie die `next()`-Methode der SQLStatement-Instanz auf. Wie die `execute()`-Methode wird der erste Parameter der `next()`-Methode verwendet, um anzuzeigen, wie viele Zeilen beim nächsten Auslösen des result-Ereignisses abgerufen werden sollen.

```
function selectResult(event:SQLEvent):void
{
    var result:SQLResult = stmt.getResult();
    if (result.data != null)
    {
        // ... loop through the rows or perform other processing ...

        if (!result.complete)
        {
            stmt.next(20); // retrieve the next 20 rows
        }
        else
        {
            stmt.removeEventListener(SQLEvent.RESULT, selectResult);
        }
    }
}
```

Die SQLStatement-Instanz löst ein `result`-Ereignis aus, wenn die `next()`-Methode eine folgende Gruppe von Ergebniszeilen zurückgibt. Deswegen kann dieselbe Listener-Funktion verwendet werden, um die Verarbeitung von Ergebnissen (aus `next()`-Aufrufen) fortzusetzen, bis alle Zeilen abgerufen wurden.

Weitere Informationen finden Sie in den Beschreibungen der `SQLStatement.execute()`-Methode (Beschreibung des `prefetch`-Parameters) und der `SQLStatement.next()`-Methode.

Einfügen von Daten

Adobe AIR 1.0 und höher

Zum Hinzufügen von Daten zu einer Datenbank muss eine SQL-Anweisung des Typs `INSERT` ausgeführt werden. Wenn die Ausführung der Anweisung abgeschlossen ist, können Sie auf den Primärschlüssel für die neu eingefügte Zeile zugreifen, falls der Schlüssel von der Datenbank generiert wurde.

Ausführen einer INSERT-Anweisung

Adobe AIR 1.0 und höher

Um einer Tabelle in einer Datenbank Daten hinzuzufügen, müssen Sie eine SQLStatement-Instanz erstellen und ausführen, deren Text eine SQL-Anweisung des Typs `INSERT` ist.

Im folgenden Beispiel wird eine SQLStatement-Instanz verwendet, um der bereits vorhandenen Tabelle „employees“ eine Datenzeile hinzuzufügen. Dieses Beispiel veranschaulicht das Hinzufügen von Daten im asynchronen Ausführungsmodus. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass eine SQLConnection-Instanz namens `conn` bereits instanziiert wurde und eine Verbindung mit der Datenbank besteht. Außerdem wird vorausgesetzt, dass die Tabelle „employees“ bereits erstellt wurde.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var insertStmt:SQLStatement = new SQLStatement();
insertStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "INSERT INTO employees (firstName, lastName, salary) " +
    "VALUES ('Bob', 'Smith', 8000)";
insertStmt.text = sql;

// register listeners for the result and failure (status) events
insertStmt.addEventListener(SQLEvent.RESULT, insertResult);
insertStmt.addEventListener(SQLErrorEvent.ERROR, insertError);

// execute the statement
insertStmt.execute();

function insertResult(event:SQLEvent):void
{
    trace("INSERT statement succeeded");
}

function insertError(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var insertStmt:SQLStatement = new SQLStatement();
                insertStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "INSERT INTO employees (firstName, lastName, salary) " +
```

```
                "VALUES ('Bob', 'Smith', 8000)";
            insertStmt.text = sql;

            // register listeners for the result and failure (status) events
            insertStmt.addEventListener(SQLEvent.RESULT, insertResult);
            insertStmt.addEventListener(SQLErrorEvent.ERROR, insertError);

            // execute the statement
            insertStmt.execute();
        }

        private function insertResult(event:SQLEvent):void
        {
            trace("INSERT statement succeeded");
        }

        private function insertError(event:SQLErrorEvent):void
        {
            trace("Error message:", event.error.message);
            trace("Details:", event.error.details);
        }
    ]]>
</mx:Script>
</mx:WindowedApplication>
```

Im folgenden Beispiel wird der bereits vorhandenen Tabelle „employees“ im synchronen Ausführungsmodus eine Datenzeile hinzugefügt. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass eine SQLConnection-Instanz namens `conn` bereits instanziiert wurde und eine Verbindung mit der Datenbank besteht. Außerdem wird vorausgesetzt, dass die Tabelle „employees“ bereits erstellt wurde.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.errors.SQLError;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var insertStmt:SQLStatement = new SQLStatement();
insertStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "INSERT INTO employees (firstName, lastName, salary) " +
    "VALUES ('Bob', 'Smith', 8000)";
insertStmt.text = sql;

try
{
    // execute the statement
    insertStmt.execute();

    trace("INSERT statement succeeded");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var insertStmt:SQLStatement = new SQLStatement();
                insertStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "INSERT INTO employees (firstName, lastName, salary) " +
                    "VALUES ('Bob', 'Smith', 8000)";
                insertStmt.text = sql;

                try
                {
                    // execute the statement
                    insertStmt.execute();
                    trace("INSERT statement succeeded");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Abrufen eines von der Datenbank generierten Primärschlüssels einer eingefügten Zeile

Adobe AIR 1.0 und höher

Häufig benötigt der Code nach dem Einfügen einer Datenzeile in eine Tabelle einen von der Datenbank generierten Primärschlüssel oder Zeilenbezeichnerwert für die neu eingefügte Zeile. Nachdem Sie eine Zeile in eine Tabelle eingefügt haben, könnten Sie zum Beispiel Zeilen in eine verwandte Tabelle einfügen. In diesem Fall würden Sie den Primärschlüssel als Fremdschlüssel in die verwandte Tabelle einfügen. Der Primärschlüssel einer neu eingefügten Zeile kann mithilfe des SQLResult-Objekts abgerufen werden, das mit der Anweisungsausführung assoziiert ist. Dies ist dasselbe Objekt, mit dem auf Ergebnisdaten zugegriffen wird, nachdem eine `SELECT`-Anweisung ausgeführt wurde. Wie bei jeder SQL-Anweisung erstellt die Laufzeitumgebung eine SQLResult-Instanz, wenn die Ausführung einer `INSERT`-Anweisung abgeschlossen ist. Sie können auf die SQLResult-Instanz zugreifen, indem Sie für das SQLStatement-Objekt die `getResult()`-Methode aufrufen, wenn Sie einen Ereignis-Listener verwenden oder im

synchronen Ausführungsmodus arbeiten. Wenn Sie im asynchronen Ausführungsmodus arbeiten und eine Responder-Instanz an den `execute()`-Aufruf übergeben, wird alternativ dazu die SQLResult-Instanz als Argument an die Ergebnisprozedurfunktion übergeben. In jedem Fall verfügt die SQLResult-Instanz über eine Eigenschaft, `lastInsertRowID`, die den Zeilenbezeichner der zuletzt eingefügten Zeile enthält, wenn die ausgeführte SQL-Anweisung eine `INSERT`-Anweisung ist.

Im folgenden Beispiel wird der Zugriff auf den Primärschlüssel einer eingefügten Zeile im asynchronen Ausführungsmodus veranschaulicht:

```
insertStmt.text = "INSERT INTO ...";

insertStmt.addEventListener(SQLEvent.RESULT, resultHandler);

insertStmt.execute();

function resultHandler(event:SQLEvent):void
{
    // get the primary key
    var result:SQLResult = insertStmt.getResult();

    var primaryKey:Number = result.lastInsertRowID;
    // do something with the primary key
}
```

Im folgenden Beispiel wird der Zugriff auf den Primärschlüssel einer eingefügten Zeile im synchronen Ausführungsmodus veranschaulicht:

```
insertStmt.text = "INSERT INTO ...";

try
{
    insertStmt.execute();

    // get the primary key
    var result:SQLResult = insertStmt.getResult();

    var primaryKey:Number = result.lastInsertRowID;
    // do something with the primary key
}
catch (error:SQLError)
{
    // respond to the error
}
```

Beachten Sie, dass der Zeilenbezeichner der Wert der Spalte sein kann, die als Primärschlüsselspalte in der Tabellendefinition bestimmt wurde; dies muss aber nicht unbedingt der Fall sein. Es gelten die folgenden Regeln:

- Wenn die Tabelle mit einer Primärschlüsselspalte definiert wurde, deren Affinität (Spaltendatentyp) `INTEGER` lautet, enthält die `lastInsertRowID`-Eigenschaft den Wert, der in diese Zeile eingefügt wurde (oder den Wert, der von der Laufzeitumgebung generiert wurde, wenn es sich um eine `AUTOINCREMENT`-Spalte handelt).
- Wenn die Tabelle mit mehreren Primärschlüsselspalten definiert wurde (ein zusammengesetzter Schlüssel) oder mit einer Primärschlüsselspalte, deren Affinität nicht `INTEGER` lautet, generiert die Datenbank im Hintergrund einen ganzzahligen Zeilenbezeichnerwert für die Zeile. Dieser generierte Wert ist der Wert der `lastInsertRowID`-Eigenschaft.

- Der Wert ist immer der Zeilenbezeichner der zuletzt eingefügten Zeile. Wenn eine `INSERT`-Anweisung einen Auslöser startet, der seinerseits eine Reihe einfügt, enthält die `lastInsertRowID`-Eigenschaft den Zeilenbezeichner der zuletzt vom Auslöser eingefügten Zeile und nicht die Zeile, die von der `INSERT`-Anweisung erstellt wurde.

Wenn Sie eine explizit definierte Primärschlüsselspalte haben möchten, deren Wert nach einem `INSERT`-Befehl über die `SQLResult.lastInsertRowID`-Eigenschaft verfügbar ist, müssen Sie die Spalte deshalb als `INTEGER PRIMARY KEY`-Spalte definieren. Selbst wenn Ihre Tabelle keine explizite `INTEGER PRIMARY KEY`-Spalte enthält, ist es dennoch möglich, den von der Datenbank generierten Zeilenbezeichner als Primärschlüssel für die Tabelle zu verwenden, wenn Sie Beziehungen zu verwandten Tabellen definieren. Der Spaltenwert des Zeilenbezeichners ist in jeder SQL-Anweisung verfügbar, indem einer der Sonderspaltennamen `ROWID`, `_ROWID_` oder `OID` verwendet wird. Sie können in einer verwandten Tabelle eine Fremdschlüsselspalte erstellen und den Zeilenbezeichnerwert als Fremdschlüsselspaltenwert verwenden, wie Sie auch mit einer explizit deklarierten `INTEGER PRIMARY KEY`-Spalte vorgehen würden. Wenn Sie einen beliebigen Primärschlüssel anstelle eines natürlichen Schlüssels verwenden und wenn es Sie nicht stört, dass die Laufzeitumgebung den Primärschlüsselwert für Sie generiert, macht es wenig Unterschied, ob Sie eine `INTEGER PRIMARY KEY`-Spalte oder den vom System generierten Zeilenbezeichner als Primärschlüssel einer Tabelle verwenden, um eine Fremdschlüsselbeziehung zwischen zwei Tabellen zu definieren.

Weitere Informationen über Primärschlüssel und generierte Zeilenbezeichner finden Sie unter „SQL-Unterstützung in lokalen Datenbanken“ auf Seite 1185.

Ändern oder Löschen von Daten

Adobe AIR 1.0 und höher

Die Vorgehensweise beim Ausführen anderer Datenbearbeitungsoperationen ist mit dem Verfahren beim Ausführen von SQL-Anweisungen des Typs `SELECT` und `INSERT` identisch, wie unter „Arbeiten mit SQL-Anweisungen“ auf Seite 777 beschrieben. Ersetzen Sie einfach die SQL-Anweisung für die SQLStatement-Instanz in der `text`-Eigenschaft:

- Um vorhandene Daten in einer Tabelle zu ändern, verwenden Sie eine `UPDATE`-Anweisung.
- Um eine oder mehrere Datenzeilen aus einer Tabelle zu löschen, verwenden Sie eine `DELETE`-Anweisung.

Beschreibungen dieser Anweisungen finden Sie unter „SQL-Unterstützung in lokalen Datenbanken“ auf Seite 1185.

Arbeiten mit mehreren Datenbanken

Adobe AIR 1.0 und höher

Mit der `SQLConnection.attach()`-Methode können Sie für eine SQLConnection-Instanz, die bereits mit einer Datenbank verbunden ist, eine Verbindung mit einer weiteren Datenbank herstellen. Sie benennen die verbundene Datenbank mit dem name-Parameter im Aufruf der `attach()`-Methode. Wenn Sie Anweisungen zur Bearbeitung dieser Datenbank schreiben, können Sie diesen Namen in einem Präfix (in der Form `Datenbankname.Tabellenname`) verwenden, um Tabellennamen in Ihren SQL-Anweisungen anzugeben. Auf diese Weise teilen Sie der Laufzeitumgebung mit, dass die Tabelle in der genannten Datenbank zu finden ist.

Sie können eine einzelne SQL-Anweisung ausführen, die Tabellen aus mehreren Datenbanken betrifft, wenn diese Datenbanken mit derselben SQLConnection-Instanz verbunden sind. Wenn eine Transaktion mit der SQLConnection-Instanz erstellt wird, gilt diese Transaktion für alle SQL-Anweisungen, die unter Verwendung dieser SQLConnection-Instanz ausgeführt werden. Dabei ist es gleichgültig, für welche der verbundenen Datenbanken die Anweisung ausgeführt wird.

Alternativ dazu können Sie auch mehrere SQLConnection-Instanzen in einer Anwendung erstellen, die jeweils mit einer Datenbank oder mit mehreren Datenbanken verbunden sind. Wenn Sie mehrere Verbindungen zu derselben Datenbank verwenden, ist zu beachten, dass eine Datenbanktransaktion nicht für alle SQLConnection-Instanzen gilt. Wenn Sie mit mehreren SQLConnection-Instanzen eine Verbindung zu derselben Datenbank herstellen, können Sie sich also nicht darauf verlassen, dass Datenänderungen in allen Verbindungen in der erwarteten Weise angewendet werden. Angenommen, zwei `UPDATE`- oder `DELETE`-Anweisungen werden über verschiedene SQLConnection-Instanzen für dieselbe Datenbank ausgeführt. Wenn nach einer Operation ein Anwendungsfehler auftritt, könnten die Datenbankdaten in einem Zwischenstadium verbleiben, das nicht rückgängig gemacht werden kann und möglicherweise die Integrität der Datenbank (und damit der Anwendung) beeinträchtigt.

Umgang mit Datenbankfehlern

Adobe AIR 1.0 und höher

Im Wesentlichen ähnelt der Umgang mit Datenbankfehlern dem Umgang mit anderen Fehlern der Laufzeitumgebung. Sie sollten Code schreiben, der auf möglicherweise auftretende Fehler vorbereitet ist und auf die Fehler reagiert, anstatt dies der Laufzeitumgebung zu überlassen. Die möglichen Datenbankfehler lassen sich in drei Kategorien aufteilen; Verbindungsfehler, SQL-Syntaxfehler und Beschränkungsfehler.

Verbindungsfehler

Adobe AIR 1.0 und höher

Bei den meisten Datenbankfehlern handelt es sich um Verbindungsfehler, die bei jeder Operation auftreten können. Es gibt zwar Strategien zur Vermeidung von Verbindungsfehlern, allerdings gibt es kaum eine Möglichkeit zum problemlosen Beheben des Verbindungsfehlers, wenn die Datenbank ein kritischer Bestandteil Ihrer Anwendung ist.

Die meisten Verbindungsfehler haben damit zu tun, wie die Laufzeitumgebung mit dem Betriebssystem, dem Dateisystem und der Datenbankdatei interagiert. Ein Verbindungsfehler tritt zum Beispiel auf, wenn der Benutzer keine Berechtigung zum Erstellen einer Datenbankdatei an einem bestimmten Speicherort im Dateisystem hat. Die folgenden Strategien tragen zur Vermeidung von Verbindungsfehlern bei:

Verwenden Sie benutzerspezifische Datenbankdateien Anstatt eine einzelne Datenbankdatei für alle Benutzer, die auf einem Computer mit der Anwendung arbeiten, zu verwenden, geben Sie jedem Benutzer eine eigene Datenbankdatei. Die Datei sollte sich in einem Verzeichnis befinden, das mit dem Benutzerkonto verknüpft ist. Dies könnte zum Beispiel im Speicherverzeichnis der Anwendung, im Dokumentordner des Benutzers oder auf dem Desktop des Benutzers sein.

Ziehen Sie verschiedene Benutzertypen in Betracht Testen Sie Ihre Anwendung mit verschiedenen Benutzertypen unter verschiedenen Betriebssystemen. Gehen Sie nicht davon aus, dass der Benutzer über Administratorberechtigungen für seinen Computer verfügt. Setzen Sie nicht voraus, dass die Person, die die Anwendung installiert, auch der Benutzer ist, der mit der Anwendung arbeitet.

Verwenden Sie verschiedene Dateispeicherorte Wenn Sie zulassen, dass Benutzer selbst festlegen, wo sie eine Datenbankdatei speichern oder eine Datei zum Öffnen auswählen, sollten Sie die möglichen Dateispeicherorte bedenken, die Benutzer verwenden können. Ziehen Sie außerdem in Erwägung, die Möglichkeiten der Benutzer einzuschränken, wenn es darum geht, Dateien zu speichern (oder wo sie Dateien öffnen können). So könnten Sie zum Beispiel festlegen, dass Benutzer nur Dateien öffnen können, die sich am Speicherort ihres Benutzerkontos befinden.

Wenn ein Verbindungsfehler auftritt, so geschieht dies meistens beim ersten Versuch, eine Datenbank zu erstellen oder zu öffnen. Dies bedeutet, dass der Benutzer keine datenbankbezogenen Operationen in der Anwendung ausführen kann. Bei bestimmten Fehlertypen, zum Beispiel Schreibschutz- oder Berechtigungsfehlern, ist ein mögliches Wiederherstellungsverfahren das Kopieren von Datenbankdateien an einen anderen Speicherort. Die

Anwendung kann die Datenbankdatei an einen anderen Speicherort kopieren, für den der Benutzer Berechtigungen zum Erstellen und zum Schreiben in Dateien hat, und dann mit diesem Speicherort arbeiten.

Syntaxfehler

Adobe AIR 1.0 und höher

Ein Syntaxfehler tritt auf, wenn eine SQL-Anweisung fehlerhaft formuliert ist und die Anwendung versucht, diese Anweisung auszuführen. Da SQL-Anweisungen für lokale Datenbanken als Strings erstellt werden, ist die Überprüfung der SQL-Syntax zur Kompilierungszeit nicht möglich. Alle SQL-Anweisungen müssen ausgeführt werden, um die Syntax zu überprüfen. Mit den folgenden Strategien können Sie SQL-Syntaxfehler vermeiden:

Testen Sie alle SQL-Anweisungen gründlich Testen Sie Ihre SQL-Anweisungen nach Möglichkeit separat, bevor Sie Anweisungstext in den Anwendungscode schreiben. Verwenden Sie zusätzlich ein Verfahren zum Testen des Codes, zum Beispiel Einheitentests, um einen Testsatz zu erstellen, der jede mögliche Option und Variation im Code ausprobiert.

Verwenden Sie Anweisungsparameter und vermeiden Sie das Verketten (dynamisches Generieren) von SQL Wenn Sie Parameter verwenden und SQL-Anweisungen nicht dynamisch generieren, wird bei jeder Ausführung einer SQL-Anweisung derselbe Anweisungstext verwendet. Deshalb ist das Testen der Anweisungen und das Beschränken der möglichen Variationen viel einfacher. Lässt sich das dynamische Generieren einer SQL-Anweisung nicht vermeiden, beschränken Sie die dynamischen Teile der Anweisung auf ein Minimum. Validieren Sie die Benutzereingaben sorgfältig, um sicherzustellen, dass sie keine Syntaxfehler verursachen.

Zur Behebung eines Syntaxfehlers benötigt eine Anwendung komplexe Logiken, um eine SQL-Anweisung zu untersuchen und ihre Syntax zu korrigieren. Indem Sie die vorstehenden Richtlinien zur Vermeidung von Syntaxfehlern befolgen, kann Ihr Code potenzielle Laufzeitquellen von SQL-Syntaxfehlern identifizieren (zum Beispiel Benutzereingaben, die in einer Anweisung verwendet werden). Geben Sie den Benutzern Anleitungen zur Wiederherstellung nach einem Syntaxfehler. Geben Sie an, welche Korrekturen erforderlich sind, damit die Anweisung korrekt ausgeführt werden kann.

Beschränkungsfehler

Adobe AIR 1.0 und höher

Beschränkungsfehler treten auf, wenn eine `INSERT`- oder `UPDATE`-Anweidung versucht, einer Spalte Daten hinzuzufügen. Zu diesem Fehler kommt es, wenn die neuen Daten eine der definierten Beschränkungen für die Tabelle oder Spalte verletzen. Mögliche Beschränkungen sind:

Eindeutigkeitsbeschränkung Gibt an, dass eine Spalte in jede Zeile der Tabelle einen anderen Wert enthalten muss. Wenn mehrere Spalten in einer Eindeutigkeitsbeschränkung zusammengefasst werden, darf die Kombination der Werte in diesen Spalten nicht doppelt vorkommen. Anders ausgedrückt: jede Zeile in der angegebenen Spalte bzw. in den angegebenen Spalten muss einmalig sein.

Primärschlüsselbeschränkung In Bezug auf die Daten, die eine Beschränkung zulässt bzw. nicht zulässt, ist eine Primärschlüsselbeschränkung identisch mit einer Eindeutigkeitsbeschränkung.

Nicht-Null-Beschränkung Legt fest, dass eine einzelne Spalte nicht den Wert `NULL` enthalten darf und deshalb in jeder Zeile einen Wert aufweisen muss.

Überprüfungsbeschränkung Ermöglicht Ihnen, eine beliebige Beschränkung für eine oder mehrere Tabellen festzulegen. Eine häufig verwendete Überprüfungsbeschränkung ist eine Regel, die festlegt, dass der Wert einer Spalte innerhalb bestimmter Grenzen liegen muss (zum Beispiel, dass der Wert in einer numerischen Spalte größer als 0 sein muss). Eine weitere gebräuchliche Überprüfungsbeschränkung legt Beziehungen zwischen Spaltenwerten fest (zum Beispiel, dass sich der Werte einer Spalte von dem einer anderen Spalte in derselben Reihe unterscheiden muss).

Datentypbeschränkung (Spaltenaffinitätsbeschränkung) Die Laufzeitumgebung erzwingt einen bestimmten Datentyp der Spaltenwerte. Wird versucht, Werte eines falschen Datentyps in einer Spalte zu speichern, tritt ein Fehler auf. In vielen Fällen werden Werte jedoch konvertiert, um zum Datentyp der Spalte zu passen. Weitere Informationen finden Sie unter „Arbeiten mit Datenbankdatentypen“ auf Seite 801.

Die Laufzeitumgebung erzwingt keine Beschränkungen für Fremdschlüsselwerte. Das bedeutet, das Fremdschlüsselwerte nicht mit einem vorhandenen Primärschlüsselwert übereinstimmen müssen.

Zusätzlich zu den vordefinierten Beschränkungstypen unterstützt die SQL-Engine der Laufzeitumgebung die Verwendung von Auslösern. Ein Auslöser ähnelt einer Ereignisprozedur. Es handelt sich um einen vordefinierten Satz von Anweisungen, die ausgeführt werden, wenn eine bestimmte Aktion eintritt. Sie können zum Beispiel einen Auslöser definieren, der ausgeführt wird, wenn Daten in eine bestimmte Tabelle eingefügt oder daraus gelöscht werden. Eine mögliche Verwendung von Auslösern ist die Untersuchung von Datenänderungen und das Ausgeben eines Fehler, wenn bestimmte Bedingungen nicht erfüllt sind. Ein Auslöser kann also demselben Zweck dienen wie eine Beschränkung und die Strategien zum Vermeiden und Beheben von Beschränkungsfehlern gelten auch für auslösergenerierte Fehler. Die Fehler-ID für auslösergenerierte Fehler unterscheidet sich jedoch von der Fehler-ID von Beschränkungsfehlern.

Den Satz der für eine bestimmte Tabelle gültigen Beschränkungen legen Sie beim Entwickeln der Anwendung fest. Wenn Sie Beschränkungen sorgfältig planen, ist die Entwicklung der Anwendung zur Vermeidung und Behebung von Beschränkungsfehlern einfacher. Es ist jedoch schwierig, Beschränkungsfehler systematisch vorherzusehen und zu verhindern. Das Vorhersehen ist schwierig, weil Beschränkungsfehler erst beim Hinzufügen von Anwendungsdaten auftreten. Beschränkungsfehler treten mit Daten auf, die einer Datenbank nach dem Erstellen hinzugefügt werden. Diese Fehler sind häufig das Ergebnis der Beziehung zwischen neuen Daten und Daten, die bereits in der Datenbank vorhanden sind. Die folgenden Strategien tragen dazu bei, viele Beschränkungsfehler zu vermeiden:

Planen Sie die Datenbankstruktur und Beschränkungen mit großer Sorgfalt Beschränkungen dienen dazu, Anwendungsregeln umzusetzen und tragen zum Schutz der Integrität der Datenbankdaten bei. Wenn Sie Ihre Anwendung planen, überlegen Sie, wie die Datenbank zu strukturieren ist, um Ihre Anwendung zu unterstützen. Identifizieren Sie im Rahmen dieses Prozesses Regeln für Ihre Daten, zum Beispiel, ob bestimmte Werte erforderlich sind, ob es Standardvorgaben für die Werte gibt, ob doppelte Werte zulässig sind usw. Orientieren Sie sich an diesen Regeln, um Datenbankbeschränkungen zu definieren.

Geben Sie Spaltennamen explizit an Eine `INSERT`-Anweisung lässt sich auch schreiben, ohne die Spalten, in die Werte eingefügt werden, ausdrücklich anzugeben; dies stellt jedoch ein unnötiges Risiko dar. Indem Sie explizit angeben, in welche Spalten Werte eingefügt werden sollen, können Sie automatisch generierte Werte zulassen, Spalten mit Standardwerten und Spalten, die `NULL`-Werte zulassen. Auf diese Weise können Sie außerdem sicherstellen, dass in alle `NOT NULL`-Spalten ein expliziter Wert eingefügt wird.

Verwenden Sie Standardwerte Wenn Sie eine `NOT NULL`-Beschränkung für eine Spalte festlegen, geben Sie nach Möglichkeit einen Standardwert in der Spaltendefinition an. Der Anwendungscode kann ebenfalls Standardwerte bereitstellen. Ihr Code kann zum Beispiel überprüfen, ob eine Stringvariable `null` ist und ihr einen Wert zuweisen, bevor sie verwendet wird, um einen Parameter in einer Anweisung festzulegen.

Validieren Sie vom Benutzer eingegebene Daten Überprüfen Sie vom Benutzer eingegebene Daten rechtzeitig, um sicherzustellen, dass durch Beschränkungen festgelegte Grenzen eingehalten werden, besonders bei `NOT NULL`- und `CHECK`-Beschränkungen. Eine `UNIQUE`-Beschränkung ist naturgemäß schwieriger zu überprüfen, da dies das Ausführen einer `SELECT`-Abfrage erfordert, um festzustellen, ob die Daten eindeutig ist.

Verwenden Sie Auslöser Sie können einen Auslöser schreiben, der eingefügte Daten validiert (und ggf. ersetzt) oder andere Aktionen ausführt, um ungültige Daten zu korrigieren. Mit dieser Überprüfung und Korrektur kann das Auftreten eines Beschränkungsfehlers verhindert werden.

Beschränkungsfehler sind in mehrerer Hinsicht schwieriger zu verhindern als andere Fehlertypen. Es gibt jedoch verschiedene Strategien, um Beschränkungsfehler zu beheben, ohne die Anwendung instabil oder unbrauchbar zu machen:

Verwenden Sie Konfliktalgorithmen Wenn Sie eine Beschränkung für eine Spalte definieren und eine `INSERT`- oder `UPDATE`-Anweisung erstellen, haben Sie die Möglichkeit, einen Konfliktalgorithmus anzugeben. Ein Konfliktalgorithmus definiert die Aktion, die die Datenbank ausführt, wenn eine Beschränkung verletzt wird. Die Datenbank-Engine kann verschiedene Aktionen ausführen. Die Datenbank-Engine kann eine einzelne Anweisung oder eine ganze Transaktion beenden. Sie kann den Fehler ignorieren. Sie kann alte Daten entfernen und durch die Daten ersetzen, die der Code zu speichern versucht.

Weitere Informationen finden Sie im Abschnitt „ON CONFLICT (Konfliktalgorithmen)“ unter „SQL-Unterstützung in lokalen Datenbanken“ auf Seite 1185.

Geben Sie korrigierende Rückmeldungen Die Beschränkungen, die einen bestimmten SQL-Befehl betreffen können, lassen sich rechtzeitig identifizieren. Dementsprechend können Sie Beschränkungsfehler, die eine Anweisung verursachen könnte, vorhersehen. Mit diesem Wissen können Sie eine Anwendungslogik erstellen, um auf Beschränkungsfehler zu reagieren. Angenommen, eine Anwendung enthält ein Dateneingabeformular, um neue Produkte einzugeben. Wenn die Produktnamenspalte in der Datenbank mit einer `UNIQUE`-Beschränkung definiert wurde, kann es beim Einfügen einer neuen Produktzeile in der Datenbank zu einem Beschränkungsfehler kommen. Die Anwendung wird deswegen so entwickelt, einen Beschränkungsfehler vorherzusehen. Wenn der Fehler auftritt, benachrichtigt die Anwendung den Benutzer, dass der angegebene Produktname bereits verwendet wird und fordert ihn auf, einen anderen Namen zu wählen. Eine andere Reaktion wäre, dem Benutzer Informationen zum Produkt mit demselben Namen anzuzeigen.

Arbeiten mit Datenbankdatentypen

Adobe AIR 1.0 und höher

Wenn in einer Datenbank eine Tabelle erstellt wird, definiert die SQL-Anweisung zum Erstellen der Tabelle die Affinität, oder den Datentyp, für jede Spalte in der Tabelle. Auch wenn Affinitätsdeklarationen ausgelassen werden können, empfiehlt es sich doch, in `CREATE TABLE`-SQL-Anweisungen die Spaltenaffinität explizit anzugeben.

Als allgemeine Regel wird jedes Objekt, das Sie mit einer `INSERT`-Anweisung in einer Datenbank speichern, als Instanz desselben Datentyps zurückgegeben, wenn Sie eine `SELECT`-Anweisung ausführen. Der Datentyp des abgerufenen Werts kann jedoch je nach Affinität der Datenbankspalte, in der der Wert gespeichert ist, ein anderer sein. Wenn ein Wert in einer Spalte gespeichert wird, und sein Datentyp nicht mit der Affinität der Spalte übereinstimmt, versucht die Datenbank, den Wert zu konvertieren. Wurde eine Datenbankspalte zum Beispiel mit der Affinität `NUMERIC` definiert, versucht die Datenbank, eingefügte Daten in eine numerische Speicherklasse zu konvertieren (`INTEGER` oder `REAL`), bevor die Daten gespeichert werden. Wenn die Daten nicht konvertiert werden können, wird ein Fehler ausgegeben. Gemäß dieser Regel wird der String „12345“, wenn er in eine `NUMERIC`-Spalte eingefügt wird, von der Datenbank automatisch in den Ganzzahlwert 12345 konvertiert, bevor er in der Datenbank gespeichert wird. Wenn der Wert mit einer `SELECT`-Anweisung abgerufen wird, wird der Wert als Instanz eines numerischen Datentyps (zum Beispiel Number) zurückgegeben, nicht als String-Instanz.

Unerwünschte Datentypkonvertierungen lassen sich am besten mit der Einhaltung von zwei Regeln vermeiden. Zuerst sollten Sie jede Spalte mit einer Affinität definieren, die dem Datentyp der zu speichernden Daten entspricht. Fügen Sie zweitens nur Werte ein, deren Datentyp der definierten Affinität entspricht. Die Einhaltung dieser Regeln hat zwei Vorteile. Wenn Sie Daten einfügen, werden sie nicht konvertiert (wobei sie die beabsichtigte Bedeutung verlieren könnten). Außerdem werden Daten mit ihrem ursprünglichen Datentyp zurückgegeben, wenn Sie sie abrufen.

Weitere Informationen zu den verfügbaren Spaltenaffinitätstypen und zur Verwendung von Datentypen in SQL-Anweisungen finden Sie unter „Unterstützte Datentypen“ auf Seite 1209.

Verwenden von synchronen und asynchronen Datenbankoperationen

Adobe AIR 1.0 und höher

In den vorangegangenen Abschnitten wurden gebräuchliche Datenbankvorgänge beschrieben, wie das Abrufen, Einfügen, Aktualisieren und Löschen von Daten, sowie das Erstellen von Datenbankdateien, Datenbanktabellen und anderen Objekten in einer Datenbank. In den Beispielen wurde veranschaulicht, wie diese Operationen im asynchronen und synchronen Ausführungsmodus erfolgen.

Im asynchronen Ausführungsmodus weisen Sie die Datenbank-Engine an, eine Operation auszuführen. Die Datenbank-Engine arbeitet dann im Hintergrund, während die Anwendung weiterhin ausgeführt wird. Nach Abschluss der Operation löst die Datenbank-Engine ein Ereignis aus, um Sie zu benachrichtigen. Der Hauptvorteil beim asynchronen Ausführungsmodus besteht darin, dass die Laufzeitumgebung Datenbankoperationen im Hintergrund ausführt, während der Hauptanwendungscode weiter ausgeführt wird. Dies ist besonders dann vorteilhaft, wenn die Ausführung der Operation viel Zeit in Anspruch nimmt.

Im synchronen Ausführungsmodus werden Operationen nicht im Hintergrund ausgeführt. Sie weisen die Datenbank-Engine an, eine Operation auszuführen. Der Code stoppt an diesem Punkt, während die Datenbank-Engine die Operation ausführt. Wenn die Operation abgeschlossen wurde, wird die Ausführung mit der nächsten Codezeile fortgesetzt.

Mit einer einzelnen Datenbankverbindung ist es nicht möglich, bestimmte Operationen oder Anweisungen synchron auszuführen, andere dagegen asynchron. Sie legen beim Öffnen der Datenbankverbindung fest, ob eine SQLConnection-Instanz synchron oder asynchron arbeitet. Wenn Sie `SQLConnection.open()` aufrufen, arbeitet die Verbindung im synchronen Ausführungsmodus; wenn Sie `SQLConnection.openAsync()` aufrufen, arbeitet die Verbindung im synchronen Ausführungsmodus. Nachdem eine SQLConnection-Instanz durch einem Aufruf von `open()` oder `openAsync()` mit einer Datenbank verbunden wurde, ist sie auf die synchrone oder asynchrone Ausführung festgelegt.

Verwenden von synchronen Datenbankoperationen

Adobe AIR 1.0 und höher

Der eigentliche Code, mit dem Operationen ausgeführt bzw. auf diese reagiert wird, unterscheidet sich im synchronen Ausführungsmodus kaum vom asynchronen Ausführungsmodus. Die Hauptunterschiede liegen in zwei Bereichen. Der erste betrifft das Ausführen einer Operation, die von einer anderen Operation abhängig ist (zum Beispiel `SELECT`-Ergebniszeilen oder der Primärschlüssel der Zeile, die mit einer `INSERT`-Anweisung hinzugefügt wurde). Der zweite Unterschied liegt im Umgang mit Fehlern.

Schreiben von Code für synchrone Operationen

Adobe AIR 1.0 und höher

Der Hauptunterschied zwischen der synchronen und asynchronen Ausführung besteht darin, dass Sie den Code im synchronen Modus als eine Reihe von Schritten schreiben. Im asynchronen Code registrieren Sie dagegen Ereignis-Listener und teilen Operationen häufig auf Listener-Methoden auf. Wenn eine Datenbank im synchronen Ausführungsmodus verbunden ist, können Sie innerhalb eines Codeblocks eine Reihe von Datenbankoperationen nacheinander ausführen. Diese Vorgehensweise wird im folgenden Beispiel veranschaulicht:

```
var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

// open the database
conn.open(dbFile, OpenMode.UPDATE);

// start a transaction
conn.begin();

// add the customer record to the database
var insertCustomer:SQLStatement = new SQLStatement();
insertCustomer.sqlConnection = conn;
insertCustomer.text =
    "INSERT INTO customers (firstName, lastName) " +
    "VALUES ('Bob', 'Jones')";
insertCustomer.execute();

var customerId:Number = insertCustomer.getResult().lastInsertRowID;

// add a related phone number record for the customer
var insertPhoneNumber:SQLStatement = new SQLStatement();
insertPhoneNumber.sqlConnection = conn;
insertPhoneNumber.text =
    "INSERT INTO customerPhoneNumbers (customerId, number) " +
    "VALUES (:customerId, '800-555-1234')";
insertPhoneNumber.parameters[":customerId"] = customerId;
insertPhoneNumber.execute();

// commit the transaction
conn.commit();
```

Sie sehen, dass Sie dieselben Methoden aufrufen, um Datenbankoperationen auszuführen, unabhängig vom Ausführungsmodus. Der Hauptunterschied zwischen den beiden Ansätzen besteht im Ausführen von Operationen, die von anderen Operationen abhängig sind, um im Umgang mit Fehlern.

Ausführen einer Operation, die von einer anderen Operation abhängig ist

Adobe AIR 1.0 und höher

Wenn Sie den synchronen Ausführungsmodus verwenden, brauchen Sie keinen Code zu schreiben, der auf ein Ereignis wartet, um festzustellen, ob eine Operation abgeschlossen ist. Sie können vielmehr davon ausgehen, dass die Ausführung der Anwendung mit der nächsten Codezeile fortgesetzt wird, wenn eine Operation in einer Codezeile erfolgreich abgeschlossen wurde. Um eine Operation auszuführen, die vom Erfolg einer anderen Operation abhängig ist, brauchen Sie den abhängigen Code also nur direkt nach der Operation, von der er abhängig ist, zu schreiben. Wenn Sie zum Beispiel eine Anwendung anweisen möchten, mit einer Transaktion zu beginnen, eine `INSERT`-Anweisung auszuführen, den Primärschlüssel der eingefügten Zeile abzurufen, diesen Primärschlüssel in eine andere Zeile einer anderen Tabelle einzufügen und die Transaktion abzuschließen, können Sie den gesamten Code als eine Reihe von Anweisungen schreiben. Diese Operationen werden im folgenden Beispiel veranschaulicht:

```
var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

// open the database
conn.open(dbFile, SQLMode.UPDATE);

// start a transaction
conn.begin();

// add the customer record to the database
var insertCustomer:SQLStatement = new SQLStatement();
insertCustomer.sqlConnection = conn;
insertCustomer.text =
    "INSERT INTO customers (firstName, lastName) " +
    "VALUES ('Bob', 'Jones')";
insertCustomer.execute();

var customerId:Number = insertCustomer.getResult().lastInsertRowID;

// add a related phone number record for the customer
var insertPhoneNumber:SQLStatement = new SQLStatement();
insertPhoneNumber.sqlConnection = conn;
insertPhoneNumber.text =
    "INSERT INTO customerPhoneNumbers (customerId, number) " +
    "VALUES (:customerId, '800-555-1234')";
insertPhoneNumber.parameters[":customerId"] = customerId;
insertPhoneNumber.execute();

// commit the transaction
conn.commit();
```

Umgang mit Fehlern im synchronen Ausführungsmodus

Adobe AIR 1.0 und höher

Im synchronen Ausführungsmodus warten Sie nicht auf ein Fehlerereignis, um festzustellen, dass eine Operation fehlgeschlagen ist. Stattdessen umgeben Sie jeden Code, der zu Fehlern führen könnte, mit einem Satz von `try..catch..finally`-Codeblöcken. Sie schließen den Fehler auslösenden Code in den `try`-Block ein. Schreiben Sie die Aktionen, die als Reaktion auf die einzelnen Fehlertypen ausgeführt werden sollen, in separate `catch`-Blöcke. Platzieren Sie den Code, den Sie unabhängig von Erfolg oder Fehlschlagen immer ausführen möchten (zum Beispiel Schließen einer nicht mehr benötigten Datenbankverbindung) in einen `finally`-Block. Im folgenden Beispiel wird die Verwendung eines `try..catch..finally`-Blocks zur Fehlerverarbeitung veranschaulicht. Es basiert auf dem vorherigen Beispiel und erweitert es, indem Fehlerverarbeitungscode hinzugefügt wird:

```
var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

// open the database
conn.open(dbFile, SQLMode.UPDATE);

// start a transaction
conn.begin();

try
{
    // add the customer record to the database
    var insertCustomer:SQLStatement = new SQLStatement();
    insertCustomer.sqlConnection = conn;
    insertCustomer.text =
        "INSERT INTO customers (firstName, lastName)" +
        "VALUES ('Bob', 'Jones')";

    insertCustomer.execute();

    var customerId:Number = insertCustomer.getResult().lastInsertRowID;

    // add a related phone number record for the customer
    var insertPhoneNumber:SQLStatement = new SQLStatement();
    insertPhoneNumber.sqlConnection = conn;
    insertPhoneNumber.text =
        "INSERT INTO customerPhoneNumbers (customerId, number)" +
        "VALUES (:customerId, '800-555-1234')";
    insertPhoneNumber.parameters[":customerId"] = customerId;

    insertPhoneNumber.execute();

    // if we've gotten to this point without errors, commit the transaction
    conn.commit();
}
catch (error:SQLError)
{
    // rollback the transaction
    conn.rollback();
}
```

Informationen zum asynchronen Ausführungsmodell

Adobe AIR 1.0 und höher

Ein häufig geäußerter Vorbehalt gegenüber dem asynchronen Ausführungsmodus ist die Annahme, dass eine SQLStatement-Instanz nicht ausgeführt werden kann, solange noch eine andere SQLStatement-Instanz für dieselbe Datenbankverbindung ausgeführt wird. Diese Annahme ist jedoch falsch. Während eine SQLStatement-Instanz ausgeführt wird, können Sie die `text`-Eigenschaft dieser Anweisung nicht ändern. Wenn Sie jedoch für jede SQL-Anweisung, die Sie ausführen möchten, eine separate SQLStatement-Instanz verwenden, können Sie die `execute()`-Methode einer SQLStatement-Instanz aufrufen, während eine andere SQLStatement-Instanz noch ausgeführt wird, ohne dass es zu einem Fehler kommt.

Intern hat beim asynchronen Ausführen von Datenbankoperationen jede Datenbankverbindung (jede SQLConnection-Instanz) ihre eigene Warteschlange oder Liste mit Operationen, die sie ausführen soll. Die Laufzeitumgebung führt die einzelnen Operationen in der Reihenfolge aus, in der sie der Warteschlange hinzugefügt wurden. Wenn Sie eine SQLStatement-Instanz erstellen und deren `execute()`-Methode aufrufen, wird die Ausführungsoperation dieser Anweisung der Warteschlange für diese Verbindung hinzugefügt. Wird zurzeit keine Operation für diese SQLConnection-Instanz ausgeführt, beginnt die Anweisung im Hintergrund mit der Ausführung. Angenommen, Sie erstellen innerhalb desselben Codeblocks eine weitere SQLStatement-Instanz und rufen deren `execute()`-Methode auf. Diese zweite Ausführungsoperation wird der Warteschlange nach der ersten Anweisung hinzugefügt. Sobald die Ausführung der ersten Anweisung abgeschlossen ist, fährt die Laufzeitumgebung mit der nächsten Operation in der Warteschlange fort. Die Verarbeitung nachfolgender Operationen in der Warteschlange erfolgt im Hintergrund, während das `result`-Ereignis für die erste Operation im Hauptanwendungscode ausgelöst wird. Diese Vorgehensweise wird im folgenden Codebeispiel veranschaulicht:

```
// Using asynchronous execution mode
var stmt1:SQLStatement = new SQLStatement();
stmt1.sqlConnection = conn;

// ... Set statement text and parameters, and register event listeners ...

stmt1.execute();

// At this point stmt1's execute() operation is added to conn's execution queue.

var stmt2:SQLStatement = new SQLStatement();
stmt2.sqlConnection = conn;

// ... Set statement text and parameters, and register event listeners ...

stmt2.execute();

// At this point stmt2's execute() operation is added to conn's execution queue.
// When stmt1 finishes executing, stmt2 will immediately begin executing
// in the background.
```

Die automatische Ausführung der Operationen in der Warteschlange durch die Datenbank hat einen wichtigen Nebeneffekt. Wenn eine Anweisung vom Ergebnis einer anderen Operation abhängig ist, können Sie die Anweisung nicht der Warteschlange hinzufügen (anders ausgedrückt, Sie können deren `execute()`-Methode nicht aufrufen), bevor die erste Operation abgeschlossen ist. Dies liegt daran, dass Sie nach dem Aufrufen der `execute()`-Methode der zweiten Anweisung die Eigenschaften `text` oder `parameters` der Anweisung nicht mehr ändern können. In diesem Fall müssen Sie auf das Ereignis, das den Abschluss der ersten Operation meldet, warten, bevor Sie mit der nächsten Operation beginnen können. Wenn Sie zum Beispiel eine Anweisung im Kontext einer Transaktion ausführen möchten, ist die Ausführung der Anweisung von der Operation abhängig, die die Transaktion öffnet. Nachdem Sie die `SQLConnection.begin()`-Methode aufgerufen haben, um mit der Transaktion zu beginnen, müssen Sie warten, bis die SQLConnection-Instanz das `begin`-Ereignis auslöst. Erst dann können Sie die `execute()`-Methode der SQLStatement-Instanz aufrufen. In diesem Beispiel ist es für die richtige Ausführung der Operationen am einfachsten, wenn Sie eine Methode erstellen, die als Listener für das `begin`-Ereignis registriert ist. Der Code, mit dem die `SQLStatement.execute()`-Methode aufgerufen wird, wird innerhalb dieser Listener-Methode platziert.

Verwenden von Verschlüsselung mit SQL-Datenbanken

Adobe AIR 1.5 und höher

Alle Adobe AIR-Anwendungen verwenden dieselbe lokale Datenbank-Engine. Dementsprechend kann jede AIR-Anwendung eine Verbindung mit einer unverschlüsselten Datenbankdatei herstellen, aus der Datenbank lesen und in diese schreiben. Ab Adobe AIR 1.5 verfügt AIR über die Möglichkeit, verschlüsselte Datenbankdateien zu erstellen bzw. eine Verbindung dazu herzustellen. Wenn Sie eine verschlüsselte Datenbank verwenden, muss eine Anwendung den richten Verschlüsselungsschlüssel bereitstellen, um eine Verbindung zur Datenbank herzustellen. Wenn der falsche Verschlüsselungschlüssel (oder kein Verschlüsselungsschlüssel) angegeben wird, kann die Anwendung keine Verbindung zur Datenbank herstellen. Deshalb kann die Anwendung keine Daten aus der Datenbank lesen und keine Daten in die Datenbank schreiben bzw. darin ändern.

Um eine verschlüsselte Datenbank zu verwenden, müssen Sie die Datenbank als verschlüsselte Datenbank erstellen. Wenn Sie eine vorhandene verschlüsselte Datenbank verwenden, können Sie eine Verbindung zu der Datenbank herstellen. Sie können auch den Verschlüsselungsschlüssel einer verschlüsselten Datenbank ändern. Abgesehen vom Erstellen der verschlüsselten Datenbank und dem Herstellen der Verbindung zu einer verschlüsselten Datenbank unterscheidet sich die Arbeit mit einer verschlüsselten Datenbank nicht von der Arbeit mit einer nicht verschlüsselten Datenbank. Insbesondere die Ausführung von SQL-Anweisungen ist identisch, unabhängig davon, ob die Datenbank verschlüsselt ist oder nicht.

Verwendungszwecke einer verschlüsselten Datenbank

Adobe AIR 1.5 und höher

Die Verschlüsselung ist immer dann sinnvoll, wenn Sie den Zugriff auf Informationen, die in einer Datenbank gespeichert sind, beschränken möchten. Die Datenbankverschlüsselung von Adobe AIR lässt sich für verschiedene Zwecke einsetzen. Einige Beispiele für Situationen, die sich für verschlüsselte Datenbanken anbieten, sind folgende:

- Ein schreibgeschützter Cachespeicher von privaten Anwendungsdaten, die von einem Server heruntergeladen werden.
- Ein lokaler Anwendungsspeicher für private Daten, die mit einem Server synchronisiert werden (Daten werden an einen Server gesendet und vom Server heruntergeladen).
- Verschlüsselte Dateien, die als Dateiformat für Dokumente verwendet werden, die in der Anwendung erstellt und bearbeitet werden. Die Dateien könnten für einen bestimmten Benutzer gedacht sein oder von allen Benutzern der Anwendung gemeinsam genutzt werden.
- Jede beliebige andere Verwendung eines lokalen Datenspeichers, wie zum Beispiel unter „Verwendungszwecke lokaler Datenbanken“ auf Seite 761 beschrieben, bei der die Daten vor Personen, die Zugriff auf den Computer oder die Datenbankdateien haben, geschützt werden sollen.

Wenn Ihnen klar ist, warum Sie eine verschlüsselte Datenbank verwenden möchten, können Sie leichter entscheiden, wie Sie Ihre Anwendung aufbauen. Insbesondere hat dies Auswirkungen darauf, wie Ihre Anwendung den Verschlüsselungsschlüssel für die Datenbank erstellt, erhält und speichert. Weitere Informationen zu diesen Aspekten finden Sie unter „Überlegungen zur Verschlüsselung von Datenbanken“ auf Seite 811.

Eine Alternativmethode zum Schutz vertraulicher Daten ist die Verwendung eines verschlüsselten lokalen Speichers. Dabei wird ein einzelner ByteArray-Wert mit einem Stringschlüssel gespeichert. Nur die AIR-Anwendung, die den Wert gespeichert hat, kann darauf zugreifen, und auch dies nur auf dem Computer, auf dem der Wert gespeichert wurde. Wenn Sie den verschlüsselten lokalen Speicher verwenden, brauchen Sie keinen eigenen Verschlüsselungsschlüssel zu erstellen. Aus diesem Grund eignet sich der verschlüsselte lokale Speicher am besten

zum einfachen Speichern eines einzelnen Werts oder einer Gruppe von Werten, die sich unkompliziert in einem ByteArray kodieren lassen. Eine verschlüsselte Datenbank eignet sich am besten für größere Datenmengen, für die eine strukturierte Datenspeicherung und Abfragefunktionen erwünscht sind. Weitere Informationen zur Verwendung des verschlüsselten lokalen Datenspeichers finden Sie unter „Verschlüsselter lokaler Speicher“ auf Seite 757.

Erstellen einer verschlüsselten Datenbank

Adobe AIR 1.5 und höher

Die Datenbankdatei muss beim Erstellen verschlüsselt werden, wenn eine verschlüsselte Datenbank verwendet werden soll. Wenn eine Datenbank unverschlüsselt erstellt wird, kann sie später nicht verschlüsselt werden. Dies gilt auch im umgekehrten Fall; eine verschlüsselte Datenbank kann später nicht zu einer unverschlüsselten gemacht werden. Bei Bedarf können Sie den Verschlüsselungsschlüssel einer verschlüsselten Datenbank ändern. Ausführliche Informationen finden Sie unter „Ändern des Verschlüsselungsschlüssels einer Datenbank“ auf Seite 810. Wenn Sie eine vorhandene nicht verschlüsselte Datenbank mit Datenbankverschlüsselung nutzen möchten, erstellen Sie eine neue verschlüsselte Datenbank und kopieren Sie die vorhandene Tabellenstruktur und die Daten in die neue Datenbank.

Das Erstellen einer verschlüsselten Datenbank ist fast identisch mit dem Erstellen einer unverschlüsselten Datenbank wie unter „Erstellen von Datenbanken“ auf Seite 766 beschrieben. Sie erstellen zunächst eine SQLConnection-Instanz, die die Verbindung zur Datenbank darstellt. Sie erstellen die Datenbank durch einen Aufruf der `open()`- oder `openAsync()`-Methode des SQLConnection-Objekts. Geben Sie dabei für den Speicherort der Datenbank eine noch nicht vorhandene Datei an. Der einzige Unterschied beim Erstellen einer verschlüsselten Datenbank besteht darin, dass Sie einen Wert für den `encryptionKey`-Parameter angeben (dies ist der fünfte Parameter der `open()`-Methode und der sechste Parameter der `openAsync()`-Methode).

Ein gültiger Wert für den `encryptionKey`-Parameter ist ein ByteArray-Objekt, das genau 16 Byte enthält.

Die folgenden Beispiele veranschaulichen die Erstellung einer verschlüsselten Datenbank. Der Einfachheit halber ist der Verschlüsselungsschlüssel in diesen Beispielen im Anwendungscode fest vorgegeben. Von dieser Methode wird jedoch dringend abgeraten, da sie nicht sicher ist.

```
var conn:SQLConnection = new SQLConnection();

var encryptionKey:ByteArray = new ByteArray();
encryptionKey.writeUTFBytes("Some16ByteString"); // This technique is not secure!

// Create an encrypted database in asynchronous mode
conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);

// Create an encrypted database in synchronous mode
conn.open(dbFile, SQLMode.CREATE, false, 1024, encryptionKey);
```

Ein Beispiel für das empfohlene Verfahren zum Generieren eines Verschlüsselungsschlüssels finden Sie unter „Beispiel: Generieren und Verwenden von Verschlüsselungsschlüsseln“ auf Seite 812.

Herstellen einer Verbindung mit einer verschlüsselten Datenbank

Adobe AIR 1.5 und höher

Wie beim Erstellen von Datenbanken gibt es auch beim Herstellen einer Verbindung zu einer unverschlüsselten bzw. zu einer verschlüsselten Datenbank große Ähnlichkeiten. Das Verfahren wird ausführlicher unter „Herstellen von Verbindungen mit Datenbanken“ auf Seite 774 beschrieben. Sie verwenden die `open()`-Methode zum Öffnen einer Verbindung im synchronen Ausführungsmodus oder die `openAsync()`-Methode zum Öffnen einer Verbindung im asynchronen Ausführungsmodus. Der einzige Unterschied besteht darin, dass Sie beim Öffnen einer verschlüsselten Datenbank den richtigen Wert für den `encryptionKey`-Parameter angeben müssen (dies ist der fünfte Parameter der `open()`-Methode und der sechste Parameter der `openAsync()`-Methode).

Wenn Sie einen falschen Verschlüsselungsschlüssel angeben, wird ein Fehler ausgegeben. Für die `open()`-Methode wird eine SQLError-Ausnahme ausgegeben. Für die `openAsync()`-Methode löst das SQLConnection-Objekt ein SQLErrorEvent-Ereignis aus, dessen `error`-Eigenschaft ein SQLError-Objekt enthält. In beiden Fällen hat das von der Ausnahme generierte SQLError-Objekt die `errorID`-Eigenschaft mit dem Wert 3138. Diese Fehler-ID entspricht der Fehlermeldung „File opened is not a database file“ (Die geöffnete Datei ist keine Datenbankdatei).

Im folgenden Beispiel wird das Öffnen einer verschlüsselten Datenbank im asynchronen Ausführungsmodus beschrieben. Der Einfachheit halber ist der Verschlüsselungsschlüssel in diesem Beispiel im Anwendungscode fest vorgegeben. Von dieser Methode wird jedoch dringend abgeraten, da sie nicht sicher ist.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);
var dbFile:File = File.applicationStorageDirectory.resolvePath("DBSample.db");

var encryptionKey:ByteArray = new ByteArray();
encryptionKey.writeUTFBytes("Some16ByteString"); // This technique is not secure!

conn.openAsync(dbFile, SQLMode.UPDATE, null, false, 1024, encryptionKey);

function openHandler(event:SQLEvent):void
{
    trace("the database opened successfully");
}

function errorHandler(event:SQLErrorEvent):void
{
    if (event.error.errorID == 3138)
    {
        trace("Incorrect encryption key");
    }
    else
    {
        trace("Error message:", event.error.message);
        trace("Details:", event.error.details);
    }
}
```

Im folgenden Beispiel wird das Öffnen einer verschlüsselten Datenbank im synchronen Ausführungsmodus beschrieben. Der Einfachheit halber ist der Verschlüsselungsschlüssel in diesem Beispiel im Anwendungscode fest vorgegeben. Von dieser Methode wird jedoch dringend abgeraten, da sie nicht sicher ist.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();
var dbFile:File = File.applicationStorageDirectory.resolvePath("DBSample.db");

var encryptionKey:ByteArray = new ByteArray();
encryptionKey.writeUTFBytes("Some16ByteString"); // This technique is not secure!

try
{
    conn.open(dbFile, SQLMode.UPDATE, false, 1024, encryptionKey);
    trace("the database was created successfully");
}
catch (error:SQLError)
{
    if (error.errorID == 3138)
    {
        trace("Incorrect encryption key");
    }
    else
    {
        trace("Error message:", error.message);
        trace("Details:", error.details);
    }
}
```

Ein Beispiel für das empfohlene Verfahren zum Generieren eines Verschlüsselungsschlüssels finden Sie unter „Beispiel: Generieren und Verwenden von Verschlüsselungsschlüsseln“ auf Seite 812.

Ändern des Verschlüsselungsschlüssels einer Datenbank

Adobe AIR 1.5 und höher

Wenn eine Datenbank verschlüsselt ist, können Sie den Verschlüsselungsschlüssel auch zu einem späteren Zeitpunkt ändern. Um den Verschlüsselungsschlüssel einer Datenbank zu ändern, öffnen Sie zunächst eine Verbindung mit der Datenbank, indem Sie eine SQLConnection-Instanz erstellen und ihre `open()`- oder `openAsync()`-Methode aufrufen. Wenn die Verbindung zur Datenbank hergestellt wurde, rufen Sie die `reencrypt()`-Methode auf und übergeben den neuen Verschlüsselungsschlüssel als Argument.

Wie bei den meisten Datenbankoperationen richtet sich das Verhalten der `reencrypt()`-Methode danach, ob die Datenbankverbindung den synchronen oder den asynchronen Ausführungsmodus verwendet. Wenn Sie die Verbindung zur Datenbank mit der `open()`-Methode herstellen, wird die `reencrypt()`-Operation synchron ausgeführt. Wenn die Operation abgeschlossen ist, wird die Ausführung mit der nächsten Codezeile fortgesetzt:

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey
conn.reencrypt(newKey);
```

Wird die Verbindung zur Datenbank dagegen mit der `openAsync()`-Methode hergestellt, wird die `reencrypt()`-Operation asynchron ausgeführt. Mit dem Aufruf von `reencrypt()` beginnt die Neuverschlüsselung. Nach Abschluss der Operation löst das SQLConnection-Objekt ein `reencrypt`-Ereignis aus. Sie verwenden einen Ereignis-Listener, um festzustellen, wann die Neuverschlüsselung abgeschlossen ist:

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey

conn.addEventListener(SQLEvent.REENCRYPT, reencryptHandler);

conn.reencrypt(newKey);

function reencryptHandler(event:SQLEvent):void
{
    // save the fact that the key changed
}
```

Die `reencrypt()`-Operation wird in einer eigenen Transaktion ausgeführt. Wenn die Operation unterbrochen wird oder fehlschlägt (zum Beispiel, weil die Anwendung vor Abschluss der Operation beendet wird), wird die Transaktion zurückgenommen. In diesem Fall ist der ursprüngliche Verschlüsselungsschlüssel weiterhin der gültige Verschlüsselungsschlüssel für die Datenbank.

Mit der `reencrypt()`-Methode kann der Verschlüsselungsschlüssel nicht von der Datenbank entfernt werden. Wenn Sie einen `null`-Wert oder einen Verschlüsselungsschlüssel, der kein 16-Byte-ByteArray ist, an die `reencrypt()`-Methode übergeben, wird ein Fehler ausgegeben.

Überlegungen zur Verschlüsselung von Datenbanken

Adobe AIR 1.5 und höher

Im Abschnitt „Verwendungszwecke einer verschlüsselten Datenbank“ auf Seite 807 sind verschiedene Fälle aufgeführt, in denen die Verwendung einer verschlüsselten Datenbank angebracht ist. Selbstverständlich gelten für unterschiedliche Verwendungsszenarien unterschiedliche Datenschutzanforderungen. Wie Sie den Einsatz von Verschlüsselung in Ihre Anwendung einbinden, spielt eine wichtige Rolle beim Schutz der Daten in einer Datenbank. Wenn Sie zum Beispiel eine verschlüsselte Datenbank verwenden, um private Daten zu schützen, auch vor Benutzern desselben Computers, benötigt die Datenbank der einzelnen Benutzer jeweils einen eigenen Verschlüsselungsschlüssel. Die höchstmögliche Sicherheit erzielen Sie, wenn die Anwendung den Schlüssel aus einem vom Benutzer eingegebenen Kennwort generieren kann. Wenn der Verschlüsselungsschlüssel auf einem Kennwort basiert, wird sichergestellt, dass auch dann kein Zugriff auf die Daten möglich ist, wenn eine andere Person das Konto des Benutzers auf dem Computer verwendet. Ein anderes Szenario liegt vor, wenn eine Datenbankdatei von allen Benutzern Ihrer Anwendung, aber nicht von anderen Anwendungen gelesen werden soll. In diesem Fall benötigt jede installierte Kopie der Anwendung Zugriff auf einen gemeinsam genutzten Verschlüsselungsschlüssel.

Sie können Ihre Anwendung und insbesondere das Verfahren zum Generieren des Verschlüsselungsschlüssels entsprechend Ihren Anforderungen für den Schutz der Anwendungsdaten entwerfen. In der folgenden Liste finden Sie verschiedene Vorschläge für unterschiedliche Stufen des Datenschutzes:

- Damit eine Datenbank für jeden Benutzer, der Zugriff auf die Anwendung hat, auf jedem Computer zugänglich ist, verwenden Sie einen einzelnen Schlüssel, der allen Instanzen der Anwendung zur Verfügung steht. Zum Beispiel kann eine Anwendung, wenn sie zum ersten Mal ausgeführt wird, den gemeinsam genutzten Verschlüsselungsschlüssel über ein sicheres Protokoll wie SSL von einem Server herunterladen. Der Schlüssel kann dann zur späteren Verwendung im verschlüsselten lokalen Speicher gespeichert werden. Alternativ dazu können Sie die Daten auf dem Computer auf Benutzerbasis verschlüsseln und die Daten mit einem Remote-Datenspeicher, zum Beispiel einem Server, synchronisieren, damit die Daten übertragbar sind.

- Damit die Daten nur für einen einzelnen Benutzer auf einem Computer zugänglich sind, generieren Sie den Verschlüsselungsschlüssel aus einer benutzerspezifischen geheimen Eingabe (zum Beispiel aus einem Kennwort). Verwenden Sie keinen Wert, der einem bestimmten Computer zugeordnet ist (zum Beispiel ein Wert, der im verschlüsselten lokalen Speicher gespeichert ist), um den Schlüssel zu generieren. Alternativ dazu können Sie die Daten auf dem Computer auf Benutzerbasis verschlüsseln und die Daten mit einem Remote-Datenspeicher, zum Beispiel einem Server, synchronisieren, damit die Daten übertragbar sind.
- Damit eine Datenbank nur einer einzelnen Person auf einem einzelnen Computer zugänglich ist, generieren Sie den Schlüssel aus einem Kennwort und einem „Salt“ (zufällig gewählte Bitfolge). Ein Beispiel für diese Vorgehensweise finden Sie unter „Beispiel: Generieren und Verwenden von Verschlüsselungsschlüsseln“ auf Seite 812.

Nachstehend sind zusätzliche Sicherheitsüberlegungen aufgeführt, die beim Entwickeln einer Anwendung zu beachten sind, die eine verschlüsselte Datenbank verwendet:

- Ein System ist immer nur so sicher wie sein schwächster Bestandteil. Wenn der Verschlüsselungsschlüssel anhand eines vom Benutzer eingegebenen Kennworts generiert wird, ziehen Sie in Betracht, Mindestlänge und Komplexitätsstufe des Kennworts festzulegen. Ein kurzes Kennwort, das nur Buchstaben oder Ziffern enthält, ist relativ leicht herauszufinden.
- Der Quellcode einer AIR-Anwendung wird als einfacher Text (für HTML-Inhalte) oder im leicht zu dekompilierenden Binärformat (für SWF-Inhalt) auf dem Computer des Benutzers gespeichert. Da der Quellcode zugänglich ist, beachten Sie diese zwei Punkte:
  - Geben Sie Verschlüsselungsschlüssel nie mit Hard-Coding im Quellcode an.
  - Gehen Sie immer davon aus, dass Angreifer das zum Generieren des Verschlüsselungsschlüssels verwendete Verfahren (zum Beispiel durch einen Zeichenzufallsgenerator oder einen bestimmten Hashing-Algorithmus) leicht herausfinden können.
- Für die AIR-Datenbankverschlüsselung wird der Advanced Encryption Standard (AES) mit Zähler im CBC-MAC-Modus (CCM) verwendet. Bei dieser Verschlüsselungsart muss ein vom Benutzer eingegebener Schlüssel mit einem Salt-Wert kombiniert werden, damit er sicher ist. Ein Beispiel für diese Vorgehensweise finden Sie unter „Beispiel: Generieren und Verwenden von Verschlüsselungsschlüsseln“ auf Seite 812.
- Wenn Sie eine Datenbank verschlüsseln, werden alle Festplattendateien verschlüsselt, die von der Datenbankengine zusammen mit dieser Datenbank verwendet werden. Die Datenbankengine speichert einige Daten jedoch in einem Arbeitsspeicher-Cache, um die Lese- und Schreibzeitleistung bei Transaktionen zu verbessern. Alle speicherresidenten Daten bleiben unverschlüsselt. Wenn ein Angreifer auf den von einer AIR-Anwendung verwendeten Arbeitsspeicher zugreifen kann, zum Beispiel durch Verwenden eines Debuggers, sind die Daten in einer geöffneten, unverschlüsselten Datenbank zugänglich.

Beispiel: Generieren und Verwenden von Verschlüsselungsschlüsseln

Adobe AIR 1.5 und höher

In dieser Beispielanwendung wird ein Verfahren zum Generieren eines Verschlüsselungsschlüssels veranschaulicht. Diese Anwendung soll die höchste Datenschutz- und Sicherheitsstufe für Benutzerdaten bieten. Ein wichtiger Aspekt beim Sichern privater Daten ist es, vom Benutzer zu verlangen, dass er jedes Mal ein Kennwort eingeben muss, wenn die Anwendung eine Verbindung zur Datenbank herstellt. Demzufolge sollte eine Anwendung, die diese Datenschutzstufe erfordert, den Verschlüsselungsschlüssel der Datenbank nie direkt speichern, wie in diesem Beispiel gezeigt.

Die Anwendung besteht aus zwei Teilen: einer ActionScript-Klasse, die einen Verschlüsselungsschlüssel generiert (die EncryptionKeyGenerator-Klasse), und einer Basisbenutzeroberfläche, die die Verwendung dieser Klasse veranschaulicht. Den vollständigen Quellcode finden Sie unter „Vollständiger Beispielcode für das Generieren und Verwenden eines Verschlüsselungsschlüssels“ auf Seite 815.

Beziehen eines sicheren Verschlüsselungsschlüssels mit der EncryptionKeyGenerator-Klasse

Adobe AIR 1.5 und höher

Ein detailliertes Verständnis der EncryptionKeyGenerator-Klasse ist nicht erforderlich, damit Sie diese Klasse in Ihrer Anwendung einsetzen können. Wenn Sie daran interessiert sind, wie die Klasse einen Verschlüsselungsschlüssel für eine Datenbank erstellt, lesen Sie „Informationen über die EncryptionKeyGenerator-Klasse“ auf Seite 819.

Gehen Sie folgendermaßen vor, um die EncryptionKeyGenerator-Klasse in Ihrer Anwendung zu verwenden:

1. Laden Sie die EncryptionKeyGenerator-Klasse als Quellcode oder eine kompilierte SWC-Datei herunter. Die EncryptionKeyGenerator-Klasse ist im Opensource-Projekt für die ActionScript 3.0-Kernbibliothek (as3corelib) enthalten. Sie können das as3corelib-Paket einschließlich Quellcode und Dokumentation herunterladen. Sie können die SWC- oder Quellcodedateien auch von der Projektseite herunterladen.
2. Legen Sie den Quellcode für die EncryptionKeyGenerator-Klasse (oder die as3corelib-SWC) an einem Speicherort ab, an dem der Quellcode Ihrer Anwendung ihn finden kann.
3. Fügen Sie in Ihrem Anwendungsquellcode eine `import`-Anweisung für die EncryptionKeyGenerator-Klasse hinzu.

   ```
   import com.adobe.air.crypto.EncryptionKeyGenerator;
   ```

4. Fügen Sie vor der Stelle, an der der Code die Datenbank erstellt oder eine Verbindung zu ihr herstellt, Code zum Erstellen einer EncryptionKeyGenerator-Instanz hinzu, indem der `EncryptionKeyGenerator()`-Konstruktor aufgerufen wird.

   ```
   var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
   ```

5. Beziehen Sie ein Kennwort vom Benutzer:

   ```
   var password:String = passwordInput.text;

   if (!keyGenerator.validateStrongPassword(password))
   {
       // display an error message
       return;
   }
   ```

   Die EncryptionKeyGenerator-Instanz verwendet dieses Kennwort als Grundlage für den Verschlüsselungsschlüssel (siehe nächster Schritt). Die EncryptionKeyGenerator-Instanz testet das Kennwort anhand bestimmter Validierungsanforderungen für sichere Kennwörter. Wenn die Validierung fehlschlägt, kommt es zu einem Fehler. Wie der Beispielcode zeigt, können Sie das Kennwort frühzeitig überprüfen, indem die `validateStrongPassword()`-Methode des EncryptionKeyGenerator-Objekts aufgerufen wird. Auf diese Weise können Sie feststellen, ob das Kennwort die Mindestanforderungen für ein sicheres Kennwort erfüllt und einen Fehler vermeiden.

6. Genieren Sie den Verschlüsselungsschlüssel aus dem Kennwort:

   ```
   var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
   ```

   Die `getEncryptionKey()`-Methode generiert den Verschlüsselungsschlüssel (ein 16-Byte-ByteArray) und gibt ihn zurück. Sie können den Verschlüsselungsschlüssel dann verwenden, um die neue verschlüsselte Datenbank zu erstellen bzw. die vorhandene Datenbank zu öffnen.

Die `getEncryptionKey()`-Methode hat einen erforderlichen Parameter; dies ist das in Schritt 5 eingeholte Kennwort.

Hinweis: *Um die höchste Stufe von Sicherheit und Datenschutz zu erhalten, muss eine Anwendung vom Benutzer verlangen, jedes Mal ein Kennwort einzugeben, wenn eine Verbindung zur Datenbank hergestellt werden soll. Speichern Sie das Benutzerkennwort oder den Datenbankverschlüsselungsschlüssel nicht direkt. Andernfalls besteht ein Sicherheitsrisiko. Stattdessen sollte eine Anwendung, wie im folgenden Beispiel, beim Herstellen einer Datenbankverbindung dasselbe Verfahren zum Ableiten des Verschlüsselungsschlüssels aus dem Kennwort verwenden wie zuvor beim Erstellen der Datenbank.*

Die `getEncryptionKey()`-Methode akzeptiert auch einen zweiten (optionalen) Parameter, den `overrideSaltELSKey`-Parameter. Der EncryptionKeyGenerator erstellt einen zufälligen Wert (*Salt* genannt), der als Teil des Verschlüsselungsschlüssels verwendet wird. Damit der Verschlüsselungsschlüssel reproduziert werden kann, wird der Saltwert im verschlüsselten lokalen Speicher (Encrypted Local Store, ELS) der AIR-Anwendung gespeichert. Standardmäßig verwendet die EncryptionKeyGenerator-Klasse einen bestimmten String als ELS-Schlüssel. Es ist zwar unwahrscheinlich, aber nicht unmöglich, dass der Schlüssel mit einem anderen von der Anwendung verwendeten Schlüssel in Konflikt steht. Anstatt den Standardschüssel zu verwenden, können Sie Ihren eigenen ELS-Schlüssel angeben. Geben Sie in diesem Fall einen benutzerdefinierten Schlüssel an, indem Sie ihn als zweiten `getEncryptionKey()`-Parameter übergeben wie hier gezeigt:

```
var customKey:String = "My custom ELS salt key";
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password, customKey);
```

7 Erstellen oder Öffnen der Datenbank

Mit einem von der `getEncryptionKey()`-Methode zurückgegebenen Verschlüsselungsschlüssel kann Ihr Code eine neue verschlüsselte Datenbank erstellen oder versuchen, eine Verbindung zu einer vorhandenen verschlüsselten Datenbank zu herzustellen. In beiden Fällen wird die `open()`- oder `openAsync()`-Methode der SQLConnection-Klasse verwendet. Dies wird unter „Erstellen einer verschlüsselten Datenbank“ auf Seite 808 und „Herstellen einer Verbindung mit einer verschlüsselten Datenbank“ auf Seite 809 beschrieben.

In diesem Beispiel wurde die Anwendung so erstellt, dass die Datenbank im asynchronen Ausführungsmodus geöffnet wird. Der Code richtet die entsprechenden Ereignis-Listener ein und ruft die `openAsync()`-Methode auf, wobei der Verschlüsselungsschlüssel als abschließendes Argument übergeben wird:

```
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, openError);

conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
```

In den Listener-Methoden entfernt der Code die Ereignis-Listener-Registrierungen. Dann wird eine Statusmeldung angezeigt, die angibt, ob die Datenbank erstellt oder geöffnet wurde oder ob ein Fehler aufgetreten ist. Der interessanteste Tel dieser Ereignisprozeduren befindet sich in der `openError()`-Methode. In dieser Methode überprüft eine `if`-Anweisung, ob die Datenbank vorhanden ist (ob also versucht wird, eine Verbindung zu einer vorhandenen Datenbank herzustellen) und ob die Fehler-ID mit der Konstante `EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID` übereinstimmt. Wenn beide Bedingungen zutreffen, bedeutet dies wahrscheinlich, dass das vom Benutzer eingegebene Kennwort falsch ist. (Es könnte jedoch auch bedeuten, dass die angegebene Datei überhaupt keine Datenbankdatei ist.) Mit dem nachstehenden Code wird die Fehler-ID überprüft:

```
if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
{
    statusMsg.text = "Incorrect password!";
}
else
{
    statusMsg.text = "Error creating or opening database.";
}
```

Den vollständigen Code für die Beispiel-Ereignis-Listener finden Sie unter „Vollständiger Beispielcode für das Generieren und Verwenden eines Verschlüsselungsschlüssels“ auf Seite 815.

Vollständiger Beispielcode für das Generieren und Verwenden eines Verschlüsselungsschlüssels

Adobe AIR 1.5 und höher

Im Folgenden ist der gesamte Code für die Beispielanwendung „Generieren und Verwenden eines Verschlüsselungsschlüssels“ aufgeführt: Der Code besteht aus zwei Teilen.

Im Beispiel wird mit der EncryptionKeyGenerator-Klasse ein Verschlüsselungsschlüssel aus einem Kennwort erstellt. Die EncryptionKeyGenerator-Klasse ist im Opensource-Projekt für die ActionScript 3.0-Kernbibliothek (as3corelib) enthalten. Sie können das as3corelib-Paket einschließlich Quellcode und Dokumentation herunterladen. Sie können die SWC- oder Quellcodedateien auch von der Projektseite herunterladen.

Flex-Beispiel

Die MXML-Datei der Anwendung enthält den Quellcode für eine einfache Anwendung, mit der eine verschlüsselte Datenbank erstellt bzw. eine Verbindung zu dieser Datenbank hergestellt wird:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
creationComplete="init();">
    <mx:Script>
        <![CDATA[
            import com.adobe.air.crypto.EncryptionKeyGenerator;

            private const dbFileName:String = "encryptedDatabase.db";

            private var dbFile:File;
            private var createNewDB:Boolean = true;
            private var conn:SQLConnection;

            // ------- Event handling -------

            private function init():void
            {
                conn = new SQLConnection();
                dbFile = File.applicationStorageDirectory.resolvePath(dbFileName);
                if (dbFile.exists)
                {
                    createNewDB = false;
                    instructions.text = "Enter your database password to open the encrypted
database.";
                    openButton.label = "Open Database";
                }
```

```
            }

            private function openConnection():void
            {
                var password:String = passwordInput.text;

                var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();

                if (password == null || password.length <= 0)
                {
                    statusMsg.text = "Please specify a password.";
                    return;
                }

                if (!keyGenerator.validateStrongPassword(password))
                {
                    statusMsg.text = "The password must be 8-32 characters long. It must
contain at least one lowercase letter, at least one uppercase letter, and at least one number
or symbol.";
                    return;
                }

                passwordInput.text = "";
                passwordInput.enabled = false;
                openButton.enabled = false;

                var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, openError);

                conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
            }

            private function openHandler(event:SQLEvent):void
            {
                conn.removeEventListener(SQLEvent.OPEN, openHandler);
                conn.removeEventListener(SQLErrorEvent.ERROR, openError);

                statusMsg.setStyle("color", 0x009900);
                if (createNewDB)
                {
                    statusMsg.text = "The encrypted database was created successfully.";
                }
                else
                {
                    statusMsg.text = "The encrypted database was opened successfully.";
                }
            }

            private function openError(event:SQLErrorEvent):void
            {
                conn.removeEventListener(SQLEvent.OPEN, openHandler);
                conn.removeEventListener(SQLErrorEvent.ERROR, openError);

                if (!createNewDB && event.error.errorID ==
```

```
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
                {
                    statusMsg.text = "Incorrect password!";
                }
                else
                {
                    statusMsg.text = "Error creating or opening database.";
                }
            }
        ]]>
    </mx:Script>
    <mx:Text id="instructions" text="Enter a password to create an encrypted database. The next
time you open the application, you will need to re-enter the password to open the database
again." width="75%" height="65"/>
    <mx:HBox>
        <mx:TextInput id="passwordInput" displayAsPassword="true"/>
        <mx:Button id="openButton" label="Create Database" click="openConnection();"/>
    </mx:HBox>
    <mx:Text id="statusMsg" color="#990000" width="75%"/>
</mx:WindowedApplication>
```

Flash Professional-Beispiel

Die FLA-Datei der Anwendung enthält den Quellcode für eine einfache Anwendung, mit der eine verschlüsselte Datenbank erstellt bzw. eine Verbindung zu dieser Datenbank hergestellt wird: Die FLA-Datei enthält vier auf der Bühne platzierte Komponenten:

Instanzname	Komponententyp	Beschreibung
`instructions`	Label	Enthält die Anweisungen, die dem Benutzer angezeigt werden
`passwordInput`	TextInput	Eingabefeld, in das der Benutzer das Kennwort eingibt
`openButton`	Button	Schaltfläche, auf die der Benutzer nach Eingabe des Kennworts klickt
`statusMsg`	Label	Zeigt Statusmeldungen (Erfolg oder Fehlschlag) an

Der Code für die Anwendung ist in einem Schlüsselbild in Bild 1 der Hauptzeitleiste definiert. Folgendes ist der Code für die Anwendung:

```
import com.adobe.air.crypto.EncryptionKeyGenerator;

const dbFileName:String = "encryptedDatabase.db";

var dbFile:File;
var createNewDB:Boolean = true;
var conn:SQLConnection;

init();

// ------- Event handling -------

function init():void
{
    passwordInput.displayAsPassword = true;
    openButton.addEventListener(MouseEvent.CLICK, openConnection);
    statusMsg.setStyle("textFormat", new TextFormat(null, null, 0x990000));

    conn = new SQLConnection();
    dbFile = File.applicationStorageDirectory.resolvePath(dbFileName);

    if (dbFile.exists)
    {
        createNewDB = false;
        instructions.text = "Enter your database password to open the encrypted database.";
        openButton.label = "Open Database";
    }
    else
    {
        instructions.text = "Enter a password to create an encrypted database. The next time
you open the application, you will need to re-enter the password to open the database again.";
        openButton.label = "Create Database";
    }
}

function openConnection(event:MouseEvent):void
{
    var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();

    var password:String = passwordInput.text;

    if (password == null || password.length <= 0)
    {
        statusMsg.text = "Please specify a password.";
        return;
    }

    if (!keyGenerator.validateStrongPassword(password))
    {
        statusMsg.text = "The password must be 8-32 characters long. It must contain at least
one lowercase letter, at least one uppercase letter, and at least one number or symbol.";
        return;
    }

    passwordInput.text = "";
    passwordInput.enabled = false;
    openButton.enabled = false;
```

```
    var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);

    conn.addEventListener(SQLEvent.OPEN, openHandler);
    conn.addEventListener(SQLErrorEvent.ERROR, openError);

    conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
}

function openHandler(event:SQLEvent):void
{
    conn.removeEventListener(SQLEvent.OPEN, openHandler);
    conn.removeEventListener(SQLErrorEvent.ERROR, openError);

    statusMsg.setStyle("textFormat", new TextFormat(null, null, 0x009900));
    if (createNewDB)
    {
        statusMsg.text = "The encrypted database was created successfully.";
    }
    else
    {
        statusMsg.text = "The encrypted database was opened successfully.";
    }
}

function openError(event:SQLErrorEvent):void
{
    conn.removeEventListener(SQLEvent.OPEN, openHandler);
    conn.removeEventListener(SQLErrorEvent.ERROR, openError);

    if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
    {
        statusMsg.text = "Incorrect password!";
    }
    else
    {
        statusMsg.text = "Error creating or opening database.";
    }
}
```

Informationen über die EncryptionKeyGenerator-Klasse

Adobe AIR 1.5 und höher

Es ist nicht notwendig, die Funktionsweise der EncryptionKeyGenerator-Klasse vollständig zu verstehen, um mit ihr einen sicheren Verschlüsselungsschlüssel für die Anwendungsdatenbank zu erstellen. Die Verwendung dieser Klasse wird unter „Beziehen eines sicheren Verschlüsselungsschlüssels mit der EncryptionKeyGenerator-Klasse“ auf Seite 813 beschrieben. Möglicherweise sind genauere Kenntnisse der von dieser Klasse verwendeten Verfahren aber nützlich. Sie möchten die Klasse vielleicht anpassen oder einige ihrer Verfahren einbauen, wenn eine andere Datenschutzstufe gewünscht wird.

Die EncryptionKeyGenerator-Klasse ist im Opensource-Projekt für die ActionScript 3.0-Kernbibliothek (as3corelib) enthalten. Sie können das as3corelib-Paket einschließlich Quellcode und Dokumentation herunterladen. Sie können die SWC- oder Quellcodedateien auch von der Projektseite herunterladen.

Wenn der Code eine EncryptionKeyGenerator-Instanz erstellt und deren `getEncryptionKey()`-Methode aufruft, werden verschiedenen Schritte ausgeführt, um sicherzustellen, dass nur berechtigte Benutzer Zugriff auf die Daten haben. Der Prozess ist identisch mit dem Generieren eines Verschlüsselungsschlüssels aus einem vom Benutzer eingegebenen Kennwort vor dem Erstellen der Datenbank bzw. mit dem Reproduzieren des Verschlüsselungsschlüssels vor dem Herstellen einer Verbindung zur Datenbank.

Erstellen und Validieren eines sicheren Kennworts

Adobe AIR 1.5 und höher

Wenn Programmcode die `getEncryptionKey()`-Methode aufruft, wird ein Kennwort als Parameter übergeben. Das Kennwort wird als Basis für den Verschlüsselungsschlüssel verwendet. Indem Informationen verwendet werden, die nur dem Benutzer bekannt sind, wird sichergestellt, dass nur der Benutzer, der das Kennwort kennt, auf die Daten in der Datenbank zugreifen kann. Selbst wenn sich ein Angreifer Zugriff auf das Benutzerkonto auf dem Computer verschafft, kann er nicht ohne Kenntnis des Kennworts auf die Datenbank zugreifen. Aus Sicherheitsgründen speichert die Anwendung das Kennwort niemals.

Der Anwendungscode erstellt eine EncryptionKeyGenerator-Instanz und ruft deren `getEncryptionKey()`-Methode auf. Dabei übergibt sie ein vom Benutzer eingegebenes Kennwort als Argument (in diesem Beispiel die Variable `password`):

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

Als ersten Schritt beim Aufrufen der `getEncryptionKey()`-Methode überprüft die EncryptionKeyGenerator-Klasse das vom Benutzer eingegebene Kennwort, um sicherzustellen, dass die Sicherheitsanforderungen für Kennwörter erfüllt werden. Die EncryptionKeyGenerator-Klasse erfordert, dass ein Kennwort 8 - 32 Zeichen enthält. Das Kennwort muss Großbuchstaben und Kleinbuchstaben sowie mindestens eine Ziffer oder ein Symbolzeichen enthalten.

Der reguläre Ausdruck, der dieses Muster überprüft, ist als Konstante mit dem Namen `STRONG_PASSWORD_PATTERN` definiert:

```
private static const STRONG_PASSWORD_PATTERN:RegExp =
/(?=^.{8,32}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/;
```

Der Code, der das Kennwort überprüft, befindet sich in der `validateStrongPassword()`-Methode der EncryptionKeyGenerator-Klasse. Der Code lautet wie folgt:

```
public function vaidateStrongPassword(password:String):Boolean
{
    if (password == null || password.length <= 0)
    {
        return false;
    }

    return STRONG_PASSWORD_PATTERN.test(password))
}
```

Intern ruft die `getEncryptionKey()`-Methode die `validateStrongPassword()`-Methode der EncryptionKeyGenerator-Klasse auf und gibt eine Ausnahme aus, wenn das Kennwort nicht gültig ist. Die `validateStrongPassword()`-Methode ist eine öffentliche Methode, sodass der Anwendungscode ein Kennwort überprüfen kann, ohne dass die `getEncryptionKey()`-Methode aufgerufen werden muss, wodurch sich Fehler vermeiden lassen.

Erweitern des Kennworts auf 256 Bit

Adobe AIR 1.5 und höher

Später im Prozess muss das Kennwort 256 Bits lang sein. Anstatt zu verlangen, dass jeder Benutzer ein Kennwort eingibt, dass genau 256 Bits (32 Zeichen) lang ist, erstellt der Code ein längeres Kennwort, indem die Zeichen des Kennworts wiederholt werden.

Die `getEncryptionKey()`-Methode ruft die `concatenatePassword()`-Methode auf, um das lange Kennwort zu erstellen.

```
var concatenatedPassword:String = concatenatePassword(password);
```

Folgendes ist der Code für die `concatenatePassword()`-Methode:

```
private function concatenatePassword(pwd:String):String
{
    var len:int = pwd.length;
    var targetLength:int = 32;

    if (len == targetLength)
    {
        return pwd;
    }

    var repetitions:int = Math.floor(targetLength / len);
    var excess:int = targetLength % len;

    var result:String = "";

    for (var i:uint = 0; i < repetitions; i++)
    {
        result += pwd;
    }

    result += pwd.substr(0, excess);

    return result;
}
```

Wenn das Kennwort weniger als 256 Zeichen enthält, verkettet der Code das Kennwort mit sich selbst, um es auf eine Länge von 256 Bits zu bringen. Dabei wird die letzte Wiederholung ggf. gekürzt, um genau 256 Bits zu erhalten.

Generieren oder Abrufen eines 256-Bit-Salt-Werts

Adobe AIR 1.5 und höher

Der nächste Schritt besteht darin, einen 256-Bit-Salt-Wert zu bekommen, der später mit dem Kennwort kombiniert wird. Unter einem *Salt* versteht man einen zufälligen Wert, der mit einem vom Benutzer eingegebenen Wert zu einem Kennwort kombiniert wird. Durch die Verwendung eines Salts mit einem Kennwort wird sichergestellt, dass die vom System verwendete Kennwort-Salt-Kombination immer ein zufälliger Wert ist, selbst wenn der Benutzer ein lexikalisches Wort oder einen bekannten Begriff als Kennwort verwendet. Diese Zufälligkeit ist ein Schutz vor Wörterbuchangriffen, bei denen Angreifer eine Wortliste verwenden, um ein Kennwort zu erraten. Außerdem ist der Salt-Wert durch das Generieren und Speichern im verschlüsselten lokalen Speicher an das Benutzerkonto auf dem Computer mit der Datenbankdatei gebunden.

Wenn die Anwendung die `getEncryptionKey()`-Methode zum ersten Mal aufruft, erstellt der Code einen zufälligen 256-Bit-Salt-Wert. Danach lädt der Code den Salt-Wert aus dem verschlüsselten lokalen Speicher.

Der Salt-Wert wird in einer Variablen mit dem Namen `salt` gespeichert. Der Code bestimmt, ob bereits ein Salt erstellt wurde, indem versucht wird, den Salt-Wert aus dem verschlüsselten lokalen Speicher zu laden:

```
var salt:ByteArray = EncryptedLocalStore.getItem(saltKey);
if (salt == null)
{
    salt = makeSalt();
    EncryptedLocalStore.setItem(saltKey, salt);
}
```

Wenn der Code einen neuen Salt-Wert erstellt, generiert die `makeSalt()`-Methode einen zufälligen 256-Bit-Wert. Da der Wert zum Schluss im verschlüsselten lokalen Speicher gespeichert wird, wird er als ByteArray generiert. Die `makeSalt()`-Methode verwendet die `Math.random()`-Methode, um den zufälligen Wert zu generieren. Die `Math.random()`-Methode kann 256 Bits nicht in einem Schritt generieren. Stattdessen verwendet der Code eine Schleife, um `Math.random()` acht Mal aufzurufen. Jedes Mal wird ein zufälliger uint-Wert zwischen 0 und 4294967295 (der höchste uint-Wert) generiert. Ein uint-Wert wird verwendet, weil uint genau 32 Bits verwendet. Indem acht uint-Werte in das ByteArray geschrieben werden, wird ein 256-Bit-Wert generiert. Folgendes ist der Code für die `makeSalt()`-Methode:

```
private function makeSalt():ByteArray
{
    var result:ByteArray = new ByteArray;

    for (var i:uint = 0; i < 8; i++)
    {
        result.writeUnsignedInt(Math.round(Math.random() * uint.MAX_VALUE));
    }

    return result;
}
```

Wenn der Code das Salt im verschlüsselten lokalen Speicher (Encrypted Local Store, ELS) speichert oder das Salt aus dem ELS abruft, benötigt er einen Stringschlüssel, unter dem das Salt gespeichert ist. Ohne Kenntnis des Schlüssels kann der Salt-Wert nicht abgerufen werden. In diesem Fall kann der Verschlüsselungsschlüssel nicht erstellt werden, um die Datenbank erneut zu öffnen. Standardmäßig verwendet die EncryptionKeyGenerator-Klasse einen vordefinierten ELS-Schlüssel, der in der Konstante `SALT_ELS_KEY` definiert ist. Anstatt den Standardschlüssel zu verwenden, kann der Anwendungscode auch einen ELS-Schlüssel angeben, der im Aufruf der `getEncryptionKey()`-Methode verwendet wird. Unabhängig davon, ob der Standardschlüssel oder ein von der Anwendung angegebener Salt-ELS-Schlüssel verwendet wird, wird der Schlüssel in einer Variablen mit dem Namen `saltKey` gespeichert. Diese Variable wird in den Aufrufen von `EncryptedLocalStore.setItem()` und `EncryptedLocalStore.getItem()` verwendet, wie bereit gezeigt.

Kombinieren des 256-Bit-Kennworts und des Salt-Werts mit dem XOR-Operator

Adobe AIR 1.5 und höher

Der Code verfügt nun über ein 256-Bit-Kennwort und einen 256-Bit-Salt-Wert. Als Nächstes wird eine XOR-Operation verwendet, um den Salt-Wert und das verlängerte Kennwort zu einem einzelnen Wert zu verbinden. Auf diese Weise wird im Endeffekt ein 256-Bit-Kennwort erstellt, das aus Zeichen besteht, die dem gesamten Bereich zulässiger Zeichen entnommen sind. Dies gilt, obwohl die eigentliche Kennworteingabe höchstwahrscheinlich hauptsächlich aus alphanumerischen Zeichen besteht. Mit dieser erhöhten Zufälligkeit wird die Gruppe der möglichen Kennwörter sehr viel größer, ohne dass der Benutzer ein langes, komplexes Kennwort eingeben muss.

Das Ergebnis der XOR-Operation wird in der Variablen `unhashedKey` gespeichert. Das eigentliche Ausführen einer bitweisen XOR-Operation für die beiden Werte erfolgt in der `xorBytes()`-Methode:

```
var unhashedKey:ByteArray = xorBytes(concatenatedPassword, salt);
```

Der bitweise XOR-Operator (^) nimmt zwei uint-Werte und gibt einen uint-Wert zurück. (Ein uint-Wert enthält 32 Bits.) Die Eingabewerte, die als Argumente an die `xorBytes()`-Methode übergeben werden, sind ein String (das Kennwort) und ein ByteArray (der Salt-Wert). Der Code verwendet eine Schleife, um jeweils 32 Bit aus jeder Eingabe zu extrahieren, um die Kombination mit dem XOR-Operator auszuführen.

```
private function xorBytes(passwordString:String, salt:ByteArray):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 4)
    {
        // ...
    }

    return result;
}
```

In der Schleife werden zunächst 32 Bit (4 Byte) aus dem `passwordString`-Parameter extrahiert. Diese Bit werden in einem zweiteiligen Prozess extrahiert und in einen uint-Wert (`o1`) konvertiert. Zuerst ruft die `charCodeAt()`-Methode den numerischen Wert der einzelnen Zeichen ab. Als Nächstes wird dieser Wert an die entsprechende Position im uint-Wert verschoben, indem der Operator für die bitweise Verschiebung nach links (`<<`) verwendet wird, und der verschobene Wert wird `o1` hinzugefügt. Das erste Zeichen (`i`) wird zum Beispiel zu den ersten 8 Bits, indem der Operator für die bitweise Verschiebung nach links (`<<`) verwendet wird, um die Bits um 24 Bits nach links zu verschieben, und dieser Wert dann `o1` zugeordnet wird. Das zweite Zeichen (`i + 1`) wird zu den zweiten 8 Bits, indem sein Wert um 16 Bits verschoben und das Ergebnis `o1` zugeordnet wird. Die Werte des dritten und vierten Zeichens werden auf dieselbe Art hinzugefügt.

```
        // ...

        // Extract 4 bytes from the password string and convert to a uint
        var o1:uint = passwordString.charCodeAt(i) << 24;
        o1 += passwordString.charCodeAt(i + 1) << 16;
        o1 += passwordString.charCodeAt(i + 2) << 8;
        o1 += passwordString.charCodeAt(i + 3);

        // ...
```

Die Variable `o1` enthält jetzt 32 Bits aus dem `passwordString`-Parameter. Als Nächstes werden 32 Bits vom `salt`-Parameter extrahiert, indem dessen `readUnsignedInt()`-Methode aufgerufen wird. Die 32 Bits werden in der uint-Variablen `o2` gespeichert.

```
        // ...

        salt.position = i;
        var o2:uint = salt.readUnsignedInt();

        // ...
```

Zum Schluss werden die beiden 32-Bit-Werte (uint-Werte) mit dem XOR-Operator kombiniert und das Ergebnis wird in einem ByteArray mit dem Namen `result` gespeichert.

```
        // ...

        var xor:uint = o1 ^ o2;
        result.writeUnsignedInt(xor);
        // ...
```

Wenn eine Schleife abgeschlossen ist, wird das ByteArray mit dem XOR-Ergebnis zurückgegeben.

```
        // ...
    }

    return result;
}
```

Hashing des Schlüssels

Adobe AIR 1.5 und höher

Nachdem das verkettete Kennwort und der Salt-Wert kombiniert wurden, besteht der nächste Schritt darin, diesen Wert noch sicherer zu gestaltet, indem Hashing mit dem SHA-256-Hashing-Algorithmus ausgeführt wird. Durch das Hashing wird Angreifern das Reverse-Engineering erschwert.

Zu diesem Zeitpunkt verfügt der Code über ein ByteArray mit den Namen `unhashedKey`, das die Kombination aus dem verketteten Kennwort und dem Salt enthält. Das ActionScript 3.0-Kernbibliothekprojekt (as3corelib) enthält eine SHA256-Klasse im com.adobe.crypto-Paket. Die `SHA256.hashBytes()`-Methode, die eine SHA-256-Hashfunktion für ein ByteArray ausführt und einen String zurückgibt, der das 256-Bit-Hash-Ergebnis als Hexadezimalzahl enthält. Die EncryptionKeyGenerator-Klasse verwendet die SHA256-Klasse für das Hashing des Schlüssels:

```
var hashedKey:String = SHA256.hashBytes(unhashedKey);
```

Extrahieren des Verschlüsselungsschlüssels aus dem Hash

Adobe AIR 1.5 und höher

Bei dem Verschlüsselungsschlüssel muss es sich um ein ByteArray handeln, das genau 16 Byte (128 Bit) lang ist. Das Ergebnis des SHA-256-Hash-Algorithmus ist immer 256 Bit lang. Der letzte Schritt besteht darin, 128 Bit aus dem Hash-Ergebnis auszuwählen, die als eigentlicher Verschlüsselungsschlüssel verwendet werden.

In der EncryptionKeyGenerator-Klasse reduziert der Code den Schlüssel auf 128 Bit, indem die `generateEncryptionKey()`-Methode aufgerufen wird. Dann wird das Ergebnis dieser Methode als Ergebnis der `getEncryptionKey()`-Methode zurückgegeben:

```
var encryptionKey:ByteArray = generateEncryptionKey(hashedKey);
return encryptionKey;
```

Es ist nicht notwendig, die ersten 128 Bits als Verschlüsselungsschlüssel zu verwenden. Sie könnten auch einen Bereich von Bits auswählen, der an einem beliebigen Punkt beginnt, Sie könnten jedes zweite Bit auswählen oder eine andere beliebige Auswahl von Bits verwenden. Wichtig ist, dass der Code 128 bestimmte Bits auswählt und jedes Mal dieselben 128 Bits verwendet.

In diesem Fall verwendet die `generateEncryptionKey()`-Methode den Bitbereich, der beim 18. Byte anfängt, als Verschlüsselungsschlüssel. Wie bereits erwähnt, gibt die SHA256-Klasse einen String zurück, der einen 256-Bit-Hash als Hexadezimalzahl enthält. Ein einzelner Block von 128 Bit hat zu viele Byte, um sie in einem Schritt dem ByteArray hinzuzufügen. Der Code verwendet eine `for`-Schleife, um Zeichen aus dem Hexidezimalstring zu extrahieren, in numerische Werte zu konvertieren und sie dem ByteArray hinzuzufügen. Der SHA-256-Ergebnisstring ist 64 Zeichen lang. Ein Bereich von 128 Bits entspricht 32 Zeichen im String, und jedes Zeichen stellt 4 Bits dar. Die kleinste Datenmenge, die Sie einem ByteArray hinzufügen können, ist ein Byte (8 Bits), was zwei Zeichen im `hash`-String entspricht. Die Schleife zählt also in Schritten von 2 Zeichen von 0 bis 31 (32 Zeichen).

In der Schleife bestimmt der Code zunächst die Startposition für das aktuelle Zeichenpaar. Da der gewünschte Bereich bei dem Zeichen an Indexposition 17 (das 18. Byte) beginnt, wird der Variable `position` der aktuelle Iteratorwert (`i`) plus 17 zugewiesen. Der Code verwendet die `substr()`-Methode des String-Objekts, um die beiden Zeichen an der aktuellen Position zu extrahieren. Diese Zeichen werden in der Variable `hex` gespeichert. Als Nächstes verwendet der Code die `parseInt()`-Methode, um den `hex`-String in einen ganzzahligen Dezimalwert zu konvertieren. Dieser Wert wird in der int-Variablen `byte` gespeichert. Zum Schluss fügt der Code den Wert von `byte` zum `result`-ByteArray hinzu, indem die `writeByte()`-Methode verwendet wird. Wenn die Schleife abgeschlossen ist, enthält das `result`-ByteArray 16 Byte und kann als Verschlüsselungsschlüssel für eine Datenbank verwendet werden.

```
private function generateEncryptionKey(hash:String):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 2)
    {
        var position:uint = i + 17;
        var hex:String = hash.substr(position, 2);
        var byte:int = parseInt(hex, 16);
        result.writeByte(byte);
    }

    return result;
}
```

Strategien für die Arbeit mit SQL-Datenbanken

Adobe AIR 1.0 und höher

Eine Anwendung kann auf verschiedene Weise mit einer lokalen SQL-Datenbank arbeiten und darauf zugreifen. Der Anwendungscode kann auf unterschiedliche Arten organisiert werden, die Reihenfolge und Zeitgebung der Ausführung von Operationen kann variiert werden usw. Die gewählten Techniken können sich auf die Entwicklung der Anwendung auswirken. Sie beeinflussen zum Beispiel, wie einfach oder komplex es ist, die Anwendung in späteren Updates zu ändern. Außerdem können sie sich auf die Leistung der Anwendung aus Benutzerperspektive auswirken.

Verteilen von vorab gefüllten Datenbanken

Adobe AIR 1.0 und höher

Wenn Sie in Ihrer AIR-Anwendung eine lokale SQL-Datenbank verwenden, erwartet die Anwendung eine Datenbank mit einer bestimmten Struktur der Tabellen, Spalten usw. Einige Anwendungen benötigen auch bestimmte bereits in die Datenbankdatei eingetragene Daten. Eine Möglichkeit, sicherzustellen, dass die Datenbank die richtige Struktur aufweist, besteht darin, die Datenbank innerhalb des Anwendungscodes zu erstellen. Beim Laden überprüft die Anwendung, ob ihre Datenbankdatei an einem bestimmten Speicherort vorhanden ist. Wenn die Datei nicht vorhanden ist, führt die Anwendung eine Reihe von Befehlen aus, um die Datenbankdatei zu erstellen, die Datenbankstruktur zu erstellen und die Tabellen mit Daten zu füllen.

Der Code, mit dem die Datenbank und ihre Tabellen erstellt werden, ist häufig komplex. Meistens wird er nur einmal im Installationszeitraum der Anwendung verwendet, aber er trägt doch dazu bei, dass Größe und Komplexität der Anwendung zunehmen. Als Alternative zum programmgesteuerten Erstellen der Datenbank, der Struktur und der Daten können Sie auch eine vorab mit Daten gefüllte Datenbank zusammen mit Ihrer Anwendung verteilen. Um eine vordefinierte Datenbank zu verteilen, schließen Sie die Datenbankdatei in das AIR-Paket der Anwendung mit ein.

Wie alle Dateien im AIR-Paket wird eine mitgelieferte Datenbankdatei im Anwendungsverzeichnis installiert (in dem Verzeichnis, dass durch die `File.applicationDirectory`-Eigenschaft angegeben wird). Dateien in diesem Verzeichnis sind jedoch schreibgeschützt. Verwenden Sie die Datei aus dem AIR-Paket deshalb als „Vorlage“ für die Datenbank. Wenn der Benutzer die Anwendung zum ersten Mal ausführt, kopieren Sie die Originaldatenbankdatei in das „Verweisen auf das Anwendungsspeicherverzeichnis“ auf Seite 716 des Benutzers (oder an einen anderen Speicherort) und verwenden Sie diese Datenbank in der Anwendung.

Empfohlene Verfahren für die Arbeit mit SQL-Datenbanken

Adobe AIR 1.0 und höher

Nachstehend sind einige Techniken aufgeführt, mit denen Sie die Leistung, die Sicherheit und die Verwaltbarkeit Ihrer Anwendung verbessern können, wenn Sie mit SQL-Datenbanken arbeiten.

Erstellen Sie Datenbankverbindungen vorab

Adobe AIR 1.0 und höher

Auch wenn Ihre Anwendung beim ersten Laden keine Anweisungen ausführt, instanziieren Sie ein SQLConnection-Objekt und rufen Sie seine `open()`- oder `openAsync()`-Methode schon im Voraus auf (zum Beispiel nach dem Starten der Anwendung), um Verzögerungen beim Ausführen von Anweisungen zu vermeiden. Lesen Sie dazu „Herstellen von Verbindungen mit Datenbanken“ auf Seite 774.

Verwenden Sie Datenbankverbindungen mehrmals

Adobe AIR 1.0 und höher

Wenn Sie während der Ausführung der Anwendung auf eine bestimmte Datenbank zugreifen, behalten Sie einen Verweis auf die SQLConnection-Instanz und verwenden Sie sie während der Ausführung der Anwendung immer wieder, anstatt die Verbindung zu schließen und wieder zu öffnen. Lesen Sie dazu „Herstellen von Verbindungen mit Datenbanken“ auf Seite 774.

Bevorzugen Sie den asynchronen Ausführungsmodus

Adobe AIR 1.0 und höher

Beim Schreiben von Code für den Datenzugriff sind manche Entwickler versucht, Operationen synchron statt asynchron auszuführen, da der Code für synchrone Operationen häufig kürzer und weniger komplex ist. Wie unter „Verwenden von synchronen und asynchronen Datenbankoperationen“ auf Seite 802 beschrieben, können synchrone Operationen die Leistung auf eine Weise beeinträchtigen, die auch vom Benutzer wahrgenommen wird und dessen Meinung über die Anwendung negativ beeinflussen kann. Wie lange die Ausführung einer einzelnen Anweisung dauert, richtet sich nach der Art der Operation und besonders nach der involvierten Datenmenge. Eine SQL-`INSERT`-Anweisung, mit der einer Datenbank nur eine einzelne Zeile hinzugefügt wird, benötigt weniger Zeit als eine `SELECT`-Anweisung, mit der Tausende von Datenzeilen abgerufen werden. Wenn Sie mehrere Operationen im synchronen Modus ausführen, sind die Operationen normalerweise verflochten. Auch wenn die einzelnen Operationen jeweils schnell ausgeführt werden, bleibt die Anwendung jedoch stehen, bis alle synchronen Operationen abgeschlossen sind. Im Ergebnis kann es insgesamt so lange dauern, bis mehrere verflochtene Operationen ausgeführt sind, dass die Anwendung zum Stillstand oder sogar Absturz kommt.

Verwenden Sie deshalb vorzugsweise asynchrone Operationen, besonders wenn die Operationen viele Datenzeilen betreffen. Wie Sie große Ergebnissätze von `SELECT`-Anweisungen aufteilen können, wird unter „Abrufen von SELECT-Ergebnissen in Teilen“ auf Seite 790 beschrieben. Dieses Verfahren kann jedoch nur im asynchronen Ausführungsmodus verwendet werden. Verwenden Sie synchrone Operationen nur dann, wenn Sie bestimmte Funktionen mit der asynchronen Programmierung nicht erreichen können, nachdem Sie die Leistungseinschränkungen für die Benutzer in Erwägung gezogen und nachdem Sie die Anwendung getestet haben, sodass Sie wissen, wie die Leistung der Anwendung betroffen ist. Der Code für die asynchrone Ausführung kann komplexer sein. Bedenken Sie jedoch, dass Sie diesen Code nur einmal schreiben müssen, die Benutzer der Anwendung ihn jedoch – schnell oder langsam – immer wieder verwenden müssen.

In einigen Fällen, wenn Sie eine separate SQLStatement-Instanz für jede auszuführende SQL-Anweisung verwenden, werden mehrere SQL-Operationen in eine Warteschlange gestellt, sodass der asynchrone Code mehr dem synchronen Code ähnelt. Lesen Sie dazu „Informationen zum asynchronen Ausführungsmodell“ auf Seite 805.

Verwenden Sie separate SQL-Anweisungen und ändern Sie nicht die text-Eigenschaft der SQLStatement-Instanz

Adobe AIR 1.0 und höher

Erstellen Sie für jede SQL-Anweisung, die in einer Anwendung mehrmals ausgeführt wird, eine separate SQLStatement-Instanz für jede SQL-Anweisung. Verwenden Sie diese SQLStatement-Instanz jedes Mal, wenn dieser SQL-Befehl ausgeführt wird. Angenommen, Sie erstellen eine Anwendung mit vier verschiedenen SQL-Operationen, die mehrmals ausgeführt werden. In diesem Fall erstellen Sie vier separate SQLStatement-Instanzen und rufen jeweils die `execute()` der Anweisung auf, um sie auszuführen. Verwenden Sie nicht eine einzelne SQLStatement-Instanz für alle SQL-Anweisungen, wobei Sie vor jedem Ausführen der Anweisung ihre `text`-Eigenschaft neu definieren müssen.

Verwenden Sie Anweisungsparameter

Adobe AIR 1.0 und höher

Arbeiten Sie mit SQLStatement-Parametern. Verwenden Sie niemals Benutzereingaben im Text der Anweisung. Sie erhöhen die Sicherheit Ihrer Anwendung, indem Sie Parameter verwenden, da so die Möglichkeit von SQL-Injection-Angriffen verhindert wird. Auf diese Weise können Sie in Abfragen Objekte verwenden (statt lediglich SQL-Literalwerte). Anweisungen werden damit außerdem effizienter ausgeführt, da sie mehrmals verwendet werden können, ohne dass sie bei jeder Ausführung neu kompiliert werden müssen. Weitere Informationen finden Sie unter „Verwenden von Parametern in Anweisungen“ auf Seite 778.

Kapitel 41: Arbeiten mit Byte-Arrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der ByteArray-Klasse können Sie Daten aus einem binären Datenstrom lesen oder in diesen schreiben. Bei einem solchen Datenstrom handelt es sich im Grunde genommen um einen Byte-Array. Die Klasse gibt Ihnen die Möglichkeit, auf Daten auf der untersten Ebene zuzugreifen. Da Computerdaten aus Byte bzw. Gruppen von je 8 Bit bestehen, bedeutet die Fähigkeit, Daten in Byte zu lesen, dass Sie auf Daten zugreifen können, für die keine Klassen und Zugriffsmethoden existieren. Mit der ByteArray-Klasse können Sie jeden beliebigen Datenstrom, von der Bitmap bis zu Datenströmen auf Netzwerken, auf Byte-Ebene analysieren.

Die Methode `writeObject()` macht es Ihnen möglich, ein Objekt im serialisierten Action Message Format (AMF) in ein ByteArray zu schreiben. Mit der Methode `readObject()` können Sie dagegen ein serialisiertes Objekt aus einem ByteArray in eine Variable des ursprünglichen Datentyps auslesen. Mit Ausnahme von Anzeigeobjekten, also Objekten, die in die Anzeigeliste aufgenommen werden können, können Sie alle Objekte serialisieren. Außerdem können Sie die serialisierten Objekte an benutzerdefinierte Klasseninstanzen zurückverweisen, wenn die benutzerdefinierte Klasse zur Laufzeit zur Verfügung steht. Wenn Sie ein Objekt in AMF konvertiert haben, können Sie es über eine Netzwerkverbindung übermitteln oder in einer Datei speichern.

Die hier beschriebene Musteranwendung von Adobe® AIR® liest eine .zip-Datei als Beispiel für die Verarbeitung eines Bytestroms. Dabei wird eine Liste der in der .zip-Datei enthaltenen Dateien extrahiert und auf den Desktop geschrieben.

Verwandte Hilfethemen

flash.utils.ByteArray

flash.utils.IExternalizable

Spezifikation des Action Message Format (AMF)

Lesen und Schreiben von ByteArrays

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die ByteArray-Klasse ist Teil des Pakets flash.utils. Erstellen Sie ein ByteArray-Objekt in ActionScript 3.0, indem Sie wie im folgenden Beispiel dargestellt die ByteArray-Klasse importieren und den Konstruktur aufrufen:

```
import flash.utils.ByteArray;
var stream:ByteArray = new ByteArray();
```

ByteArray-Methoden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jeder aussagekräftige Datenstrom ist in einem Format angeordnet, das Sie analysieren können, um die gewünschten Informationen zu finden. Ein Datensatz in einer einfachen Arbeitnehmerdatei enthält zum Beispiel in der Regel eine Kennnummer, einen Namen, eine Adresse, eine Telefonnummer und weitere Daten. Eine MP3-Audiodatei enthält einen ID3-Tag, der Titel, Autor, Album, Veröffentlichungsdatum und Genre der herunterzuladenden Datei angibt. Anhand des Formats wissen Sie, in welcher Reihenfolge Sie die Daten im Datenstrom zu erwarten haben. So können Sie den Bytestrom intelligent lesen.

Die ByteArray-Klasse umfasst mehrere Methoden, die das Lesen von und Schreiben in einem Datenstrom erleichtern. Zu diesen Methoden gehören `readBytes()` und `writeBytes()`, `readInt()` und `writeInt()`, `readFloat()` und `writeFloat()`, `readObject()` und `writeObject()` und `readUTFBytes()` und `writeUTFBytes()`. Mit diesen Methoden können Sie Daten vom Datenstrom in Variablen bestimmter Datentypen lesen und von bestimmten Datentypen direkt in den binären Datenstrom schreiben.

Der folgende Code liest zum Beispiel ein einfaches Array von Strings und Gleitkommazahlen und schreibt jedes Element in ein ByteArray. Der Aufbau des Arrays erlaubt es dem Code, die entsprechende ByteArray-Methode aufzurufen (`writeUTFBytes()` und `writeFloat()`), um die Daten zu schreiben. Aufgrund des sich wiederholenden Datenmusters kann das Array mit einer Schleife gelesen werden.

```
 // The following example reads a simple Array (groceries), made up of strings
// and floating-point numbers, and writes it to a ByteArray.

import flash.utils.ByteArray;

// define the grocery list Array
var groceries:Array = ["milk", 4.50, "soup", 1.79, "eggs", 3.19, "bread" , 2.35]
// define the ByteArray
var bytes:ByteArray = new ByteArray();
// for each item in the array
for (var i:int = 0; i < groceries.length; i++) {
        bytes.writeUTFBytes(groceries[i++]); //write the string and position to the next item
        bytes.writeFloat(groceries[i]);// write the float
        trace("bytes.position is: " + bytes.position);//display the position in ByteArray
}
trace("bytes length is: " +  bytes.length);// display the length
```

Die Eigenschaft „position“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Eigenschaft „position“ wird die aktuelle Position des Zeigers gespeichert, der das ByteArray während des Lese- oder Schreibvorgangs indiziert. Der Anfangswert der Eigenschaft ist wie im folgenden Code dargestellt 0 (Null):

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
```

Wenn Sie von einem ByteArray lesen oder in es schreiben, aktualisiert die verwendete Methode die Eigenschaft „position“ so, dass sie auf die Stelle direkt hinter dem letzten Byte verweist, das gelesen oder geschrieben wurde. Der folgende Code schreibt zum Beispiel einen String in ein ByteArray und die Eigenschaft „position“ verweist danach auf das Byte direkt nach dem String im ByteArray:

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
bytes.writeUTFBytes("Hello World!");
trace("bytes.position is now: " + bytes.position);// 12
```

Bei einem Lesevorgang wird die Eigenschaft „position“ ebenfalls um die Anzahl der gelesenen Byte erhöht.

```
var bytes:ByteArray = new ByteArray();

trace("bytes.position is initially: " + bytes.position); // 0
bytes.writeUTFBytes("Hello World!");
trace("bytes.position is now: " + bytes.position);// 12
bytes.position = 0;
trace("The first 6 bytes are: " + (bytes.readUTFBytes(6)));//Hello
trace("And the next 6 bytes are: " + (bytes.readUTFBytes(6)));// World!
```

Beachten Sie, dass Sie die Eigenschaft „position“ auf eine bestimmte Stelle im ByteArray setzen können, um ab dieser Stelle zu lesen oder zu schreiben.

Die Eigenschaften „bytesAvailable“ und „length“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Eigenschaften `length` und `bytesAvailable` geben an, wie lang ein ByteArray ist und wie viele Byte ab der aktuellen Position bis zum Ende verbleiben. Im folgenden Beispiel wird gezeigt, wie Sie diese Eigenschaften einsetzen können. Im Beispiel wird ein Textstring in das ByteArray geschrieben und das ByteArray dann Byte für Byte gelesen, bis der Buchstabe „a“ oder das Ende erreicht wird (`bytesAvailable <= 0`).

```
var bytes:ByteArray = new ByteArray();
var text:String = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vivamus etc.";

bytes.writeUTFBytes(text); // write the text to the ByteArray
trace("The length of the ByteArray is: " + bytes.length);// 70
bytes.position = 0; // reset position
while (bytes.bytesAvailable > 0 && (bytes.readUTFBytes(1) != 'a')) {
    //read to letter a or end of bytes
}
if (bytes.position < bytes.bytesAvailable) {
    trace("Found the letter a; position is: " + bytes.position); // 23
    trace("and the number of bytes available is: " + bytes.bytesAvailable);// 47
}
```

Die Eigenschaft „endian“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Verschiedene Computer verwenden verschiedene Verfahren für das Speichern aus mehreren Byte bestehender Zahlen, also Zahlen, die mehr als ein Byte Speicherplatz benötigen. Eine Ganzzahl kann beispielsweise 4 Byte, oder 32 Bit, Speicherplatz beanspruchen. Einige Computer speichern das höchstwertigste Byte der Zahl zuerst, an der niedrigsten Speicheraddresse, andere Computer speichern das niedrigwertigste Byte zuerst. Dieses Attribut der Computer, die Byte-Reihenfolge, wird als *big endian* (höchstwertigstes Byte zuerst) oder *little endian* (niedrigwertigstes Byte zuerst) bezeichnet. Die Zahl 0x31323334 wird nach den Systemen big endian und little endian wie folgt gespeichert, wobei a0 für die niedrigste Speicheradresse der 4 Byte steht und a3 für die höchste:

Big Endian	**Big Endian**	**Big Endian**	**Big Endian**
a0	a1	a2	a3
31	32	33	34

Little Endian	**Little Endian**	**Little Endian**	**Little Endian**
a0	a1	a2	a3
34	33	32	31

Mit der Eigenschaft `endian` der ByteArray-Klasse können Sie diese Byte-Reihenfolge für aus mehreren Byte bestehende Zahlen, die Sie verarbeiten, angeben. Die anerkannten Werte für diese Eigenschaft lauten entweder `„bigEndian"` oder `„littleEndian"` und die Endian-Klasse definiert die Konstanten `BIG_ENDIAN` und `LITTLE_ENDIAN` für die Einstellung der Eigenschaft `endian` mit diesen Strings.

Die Methoden „compress()“ und „uncompress()“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Methode `compress()` können Sie ein ByteArray entsprechend eines als Parameter angegebenen Komprimierungsalgorithmus komprimieren. Mit der Methode `uncompress()` dagegen können Sie ein komprimiertes ByteArray entsprechend eines als Parameter angegebenen Komprimierungsalgorithmus dekomprimieren. Nach Aufruf von `compress()` und `uncompress()` werden die Länge des Byte-Arrays auf die neue Länge und die Eigenschaft „position“ auf das Ende gesetzt.

Die CompressionAlgorithm-Klasse definiert Konstanten, mit denen Sie den Komprimierungsalgorithmus angeben können. Die ByteArray-Klasse unterstützt die Algorithmen deflate (nur AIR), zlib und lzma. Eine Beschreibung des komprimierten Datenformats zlib finden Sie unter http://www.ietf.org/rfc/rfc1950.txt. Der lzma-Algorithmus wurde für Flash Player 11.4 und AIR 3.4 hinzugefügt. Eine Beschreibung finden Sie unter http://www.7-zip.org/7z.html.

Der Komprimierungsalgorithmus „deflate“ wird in verschiedenen Komprimierungsformaten wie zlib, gzip und einigen zip-Implementationen verwendet. Eine Beschreibung des Komprimierungsalgorithmus deflate finden Sie unter http://www.ietf.org/rfc/rfc1951.txt.

Im folgenden Beispiel wird ein ByteArray mit dem Namen `bytes` mithilfe des lzma-Algorithmus komprimiert:

```
bytes.compress(CompressionAlgorithm.LZMA);
```

Im folgenden Beispiel wird ein komprimiertes ByteArray mit dem Algorithmus „deflate“ dekomprimiert:

```
bytes.uncompress(CompressionAlgorithm.LZMA);
```

Lesen und Schreiben von Objekten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Methoden `readObject()` und `writeObject()` lesen ein Objekt aus und schreiben es im serialisierten Action Message Format (AMF) in ein ByteArray. AMF ist ein herstellerspezifisches Meldungsprotokoll von Adobe, das in verschiedenen ActionScript 3.0-Klassen, einschließlich Netstream, NetConnection, NetStream, LocalConnection und Shared Objects, eingesetzt wird.

Eine aus einem Byte bestehende Markierung gibt den Typ der kodierten Daten an, die folgen. AMF verwendet die folgenden 13 Datentypen:

```
value-type = undefined-marker | null-marker | false-marker | true-marker | integer-type |
    double-type | string-type | xml-doc-type | date-type | array-type | object-type |
    xml-type | byte-array-type
```

Die kodierten Daten folgen der Typmarkierung, es sei denn, die Markierung steht für einen einzigen möglichen Wert, wie null, true oder false. In diesem Fall sind sonst keine Daten kodiert.

Es stehen zwei AMF-Versionen zur Verfügung: AMF 0 und AMF 3. AMF 0 unterstützt die Sendung komplexer Objekte durch Referenzen und lässt die Wiederherstellung von Objektbeziehungen durch Endpunkte zu. AMF 3 stellt gegenüber AMF 0 eine Verbesserung dar, da zusätzlich zu den Objektverweisen Objektmerkmale und Strings durch Referenzen gesendet werden können und in ActionScript 3.0 eingeführte neue Datentypen unterstützt werden. Die Eigenschaft `ByteArray.objectEcoding` gibt die AMF-Version an, die zur Kodierung der Objektdaten verwendet wird. Die Klasse flash.net.ObjectEncoding legt die Konstanten für die Angabe der AMF-Version fest: `ObjectEncoding.AMF0` und `ObjectEncoding.AMF3`.

Im folgenden Beispiel wird `writeObject()` aufgerufen, um ein XML-Objekt in ein ByteArray zu schreiben. Dieses wird dann mit dem deflate-Algorithmus komprimiert und in die Datei `order` auf dem Desktop geschrieben. Nach Abschluss des Vorgangs wird im AIR-Fenster die Meldung „Wrote order file to desktop!“ eingeblendet.

```
import flash.filesystem.*;
import flash.display.Sprite;
import flash.display.TextField;
import flash.utils.ByteArray;
public class WriteObjectExample extends Sprite
{
    public function WriteObjectExample()
    {
        var bytes:ByteArray = new ByteArray();
        var myLabel:TextField = new TextField();
        myLabel.x = 150;
        myLabel.y = 150;
        myLabel.width = 200;
        addChild(myLabel);

        var myXML:XML =
            <order>
                <item id='1'>
                    <menuName>burger</menuName>
                    <price>3.95</price>
                </item>
                <item id='2'>
                    <menuName>fries</menuName>
                    <price>1.45</price>
                </item>
            </order>;

        // Write XML object to ByteArray
```

```
        bytes.writeObject(myXML);
        bytes.position = 0;//reset position to beginning
        bytes.compress(CompressionAlgorithm.DEFLATE);// compress ByteArray
        writeBytesToFile("order.xml", bytes);
        myLabel.text = "Wrote order file to desktop!";
    }

    private function writeBytesToFile(fileName:String, data:ByteArray):void
    {
        var outFile:File = File.desktopDirectory; // dest folder is desktop
        outFile = outFile.resolvePath(fileName);  // name of file to write
        var outStream:FileStream = new FileStream();
        // open output file stream in WRITE mode
        outStream.open(outFile, FileMode.WRITE);
        // write out the file
        outStream.writeBytes(data, 0, data.length);
        // close it
        outStream.close();
    }
}
```

Die Methode `readObject()` liest ein Objekt im serialisierten AMF aus dem ByteArray und speichert es in einem Objekt des angegebenen Typs. Im folgenden Beispiel wird die Datei `order` aus dem Desktop in ein ByteArray (`inBytes`) gelesen, dekomprimiert und es wird `readObject()` aufgerufen, um die Daten im XML-Objekt `orderXML` zu speichern. Es wird das Schleifenkonstrukt `for each()` verwendet, um jeden Knoten für die Anzeige zu einem Textbereich hinzuzufügen. Das Beispiel zeigt auch den Wert der Eigenschaft `objectEncoding` sowie einen Header für die Inhalte der Datei `order` an.

```
import flash.filesystem.*;
import flash.display.Sprite;
import flash.display.TextField;
import flash.utils.ByteArray;

public class ReadObjectExample extends Sprite
{
    public function ReadObjectExample()
    {
        var inBytes:ByteArray = new ByteArray();
        // define text area for displaying XML content
        var myTxt:TextField = new TextField();
        myTxt.width = 550;
        myTxt.height = 400;
        addChild(myTxt);
        //display objectEncoding and file heading
        myTxt.text = "Object encoding is: " + inBytes.objectEncoding + "\n\n" + "order file: \n\n";
        readFileIntoByteArray("order", inBytes);

        inBytes.position = 0; // reset position to beginning
        inBytes.uncompress(CompressionAlgorithm.DEFLATE);
        inBytes.position = 0;//reset position to beginning
        // read XML Object
        var orderXML:XML = inBytes.readObject();
```

```
        // for each node in orderXML
        for each (var child:XML in orderXML)
        {
            // append child node to text area
            myTxt.text += child + "\n";
        }
    }

    // read specified file into byte array
    private function readFileIntoByteArray(fileName:String, data:ByteArray):void
    {
        var inFile:File = File.desktopDirectory; // source folder is desktop
        inFile = inFile.resolvePath(fileName);  // name of file to read
        var inStream:FileStream = new FileStream();
        inStream.open(inFile, FileMode.READ);
        inStream.readBytes(data);
        inStream.close();
    }
}
```

ByteArray-Beispiel: Lesen von .zip-Dateien

Adobe AIR 1.0 und höher

In diesem Beispiel wird gezeigt, wie einfache .zip-Dateien, die mehrere Dateien unterschiedlicher Dateitypen enthalten, gelesen werden. Hierfür werden die relevanten Daten aus den Metadaten jeder Datei extrahiert, jede Datei wird in ein ByteArray dekomprimiert und die Datei wird auf den Desktop geschrieben.

Die allgemeine Struktur von .zip-Dateien baut auf den Vorgaben von PKWARE Inc. auf, die Sie unter http://www.pkware.com/documents/casestudies/APPNOTE.TXT finden. Zuerst kommen der Datei-Header und die Dateidaten für die erste Datei im .zip-Archiv, gefolgt vom Datei-Header und den Daten der folgenden Dateien. (Die Struktur des Headers wird weiter unten beschrieben.) Als Nächstes enthält die .zip-Datei einen optionalen Datendeskriptordatensatz (in der Regel, wenn die .zip-Datei im Arbeitsspeicher erstellt und nicht auf einem Datenträger gespeichert wurde). Dem folgen verschiedene weitere optionale Elemente: archive decryption header, archive extra data record, central directory structure, Zip64 end of central directory record, Zip64 end of central directory locator und end of central directory record.

Der Code in diesem Beispiel wurde so geschrieben, dass nur .zip-Dateien, die keine Ordner enthalten, analysiert werden. Datendeskriptordatensätze werden nicht erwartet. Sämtliche Informationen, die auf die letzten Dateidaten folgen, werden ignoriert.

Der Datei-Header der einzelnen Dateien folgt dem Format:

Datei-Header-Signatur	4 Byte
Erforderliche Version	2 Byte
Allgemeines Bit-Flag	2 Byte
Komprimierungsverfahren	2 Byte (8=DEFLATE; 0=UNCOMPRESSED)
Letzte Änderung Uhrzeit	2 Byte
Letzte Änderung Datum	2 Byte

crc-32	4 Byte
Komprimierte Dateigröße	4 Byte
Entkomprimierte Dateigröße	4 Byte
Dateinamenlänge	2 Byte
Zusatzfeldlänge	2 Byte
Dateiname	Variable
Feldlänge	Variable

Auf den Datei-Header folgen die eigentlichen Dateidaten, die je nach Komprimierungsverfahren-Flag komprimiert oder entkomprimiert sein können. Das Flag steht auf 0 (Null), wenn die Datei nicht komprimiert ist, auf 8, wenn die Daten mit dem deflate-Algorithmus komprimiert wurden, und auf einem anderen Wert für andere Komprimierungsalgorithmen.

Die Benutzeroberfläche für dieses Beispiel besteht aus einer Beschriftung und einem Textbereich (`taFiles`). Die Anwendung schreibt die folgenden Informationen in den Textbereich jeder Datei, die sie in der .zip-Datei findet: Dateiname, komprimierte Dateigröße und entkomprimierte Dateigröße. Das folgende MXML-Dokument definiert die Benutzeroberfläche für die Flex-Version der Anwendung:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
creationComplete="init();">
    <mx:Script>
        <![CDATA[
            // The application code goes here
        ]]>
    </mx:Script>
    <mx:Form>
        <mx:FormItem label="Output">
            <mx:TextArea id="taFiles" width="320" height="150"/>
        </mx:FormItem>
    </mx:Form>
</mx:WindowedApplication>
```

Zu Beginn des Programms werden folgende Aufgaben ausgeführt:

- Importieren der erforderlichen Klassen

```
import flash.filesystem.*;
import flash.utils.ByteArray;
import flash.events.Event;
```

- Definition der Benutzeroberfläche für Flash

```
import fl.controls.*;

//requires TextArea and Label components in the Library
var taFiles = new TextArea();
var output = new Label();
taFiles.setSize(320, 150);
taFiles.move(10, 30);
output.move(10, 10);
output.width = 150;
output.text = "Contents of HelloAir.zip";
addChild(taFiles);
addChild(output);
```

- Definition des ByteArrays `bytes`

```
var bytes:ByteArray = new ByteArray();
```

- Definition der Variablen für das Speichern von Metadaten aus dem Datei-Header

```
// variables for reading fixed portion of file header
var fileName:String = new String();
var flNameLength:uint;
var xfldLength:uint;
var offset:uint;
var compSize:uint;
var uncompSize:uint;
var compMethod:int;
var signature:int;
```

- Definition der File-Objekte (`zfile`) und FileStream-Objekte (`zStream`) für die Darstellung der .zip-Datei und Angabe des Speicherorts der .zip-Datei, aus der die Dateien extrahiert werden - eine Datei mit dem Namen „HelloAIR.zip" im Desktop-Verzeichnis.

```
// File variables for accessing .zip file
var zfile:File = File.desktopDirectory.resolvePath("HelloAIR.zip");
var zStream:FileStream = new FileStream();
```

In Flex beginnt der Programmcode mit der Methode `init()`, die als `creationComplete`-Prozedur für das Stamm-Tag `mx:WindowedApplication` aufgerufen wird.

```
// for Flex
private function init():void
{
```

Das Programm beginnt mit dem Öffnen der .zip-Datei im Lesemodus.

```
zStream.open(zfile, FileMode.READ);
```

Es setzt dann die Eigenschaft `endian` von `bytes` auf `LITTLE_ENDIAN`, um festzulegen, dass bei der Byte-Reihenfolge numerischer Felder das niederwertigste Byte zuerst angegeben wird.

```
bytes.endian = Endian.LITTLE_ENDIAN;
```

Dann startet eine `while()`-Anweisung eine Schleife, die fortgesetzt wird, bis die aktuelle Position im Datenstrom der Dateigröße entspricht oder diese überschreitet.

```
while (zStream.position < zfile.size)
{
```

Die erste Anweisung in der Schleife liest die ersten 30 Byte des Dateistroms in das ByteArray `bytes`. Diese ersten 30 Byte bilden den ersten Datei-Header, dessen Größe vorgegeben ist.

```
// read fixed metadata portion of local file header
zStream.readBytes(bytes, 0, 30);
```

Danach liest der Code eine Ganzzahl (`signature`) aus den ersten Byte des 30-Byte-Headers. Durch die ZIP-Formatdefinition wird für die Signatur jedes Datei-Headers der Hexadezimalwert `0x04034b50` vorgegeben. Weicht die Signatur hiervon ab, ist der Code über den Dateiteil der .zip-Datei hinausgekommen und es sind keine weiteren Dateien zu extrahieren. In diesem Fall verlässt der Code die Schleife `while` sofort und wartet nicht auf das Ende des Byte-Arrays.

```
bytes.position = 0;
signature = bytes.readInt();
// if no longer reading data files, quit
if (signature != 0x04034b50)
{
    break;
}
```

Der nächste Abschnitt des Codes liest den Header-Byte an Offset-Position 8 und speichert den Wert in der Variablen `compMethod`. Dieses Byte enthält einen Wert, der das Komprimierungsverfahren angibt, das zur Komprimierung der Datei angewendet wurde. Es sind verschiedene Komprimierungsverfahren zulässig, praktisch verwenden jedoch nahezu alle .zip-Dateien den Komprimierungsalgorithmus „deflate“. Wurde die aktuelle Datei mit „deflate“ komprimiert, ist `compMethod` 8; ist die Datei nicht komprimiert, ist `compMethod` 0.

```
bytes.position = 8;
compMethod = bytes.readByte();  // store compression method (8 == Deflate)
```

Den ersten 30 Byte folgt ein Header-Bestandteil von variabler Länge, der den Dateinamen und möglicherweise ein zusätzliches Feld enthält. In der Variablen `offset` ist die Größe dieses Bestandteils gespeichert. Die Größe wird durch Addition der Dateinamenlänge und der Länge des Zusatzfelds berechnet, die vom Header an den Offsets 26 und 28 eingelesen werden.

```
offset = 0;// stores length of variable portion of metadata
bytes.position = 26;  // offset to file name length
flNameLength = bytes.readShort();// store file name
offset += flNameLength; // add length of file name
bytes.position = 28;// offset to extra field length
xfldLength = bytes.readShort();
offset += xfldLength;// add length of extra field
```

Als Nächstes liest das Programm den Bestandteil des Datei-Headers mit variabler Länge ein, der die Anzahl der in der Variablen `offset` gespeicherten Byte angibt.

```
// read variable length bytes between fixed-length header and compressed file data
zStream.readBytes(bytes, 30, offset);
```

Das Programm liest den Dateinamen aus dem Bestandteil des Datei-Headers mit variabler Länge und zeigt diesen im Textbereich zusammen mit der komprimierten (gezippten) und entkomprimierten (ursprünglichen) Größe der Datei an.

```
// Flash version
        bytes.position = 30;
        fileName = bytes.readUTFBytes(flNameLength); // read file name
        taFiles.appendText(fileName + "\n"); // write file name to text area
        bytes.position = 18;
        compSize = bytes.readUnsignedInt();  // store size of compressed portion
        taFiles.appendText("\tCompressed size is: " + compSize + '\n');
        bytes.position = 22;  // offset to uncompressed size
        uncompSize = bytes.readUnsignedInt();  // store uncompressed size
        taFiles.appendText("\tUncompressed size is: " + uncompSize + '\n');
```

```
// Flex version
        bytes.position = 30;
        fileName = bytes.readUTFBytes(flNameLength); // read file name
        taFiles.text += fileName + "\n"; // write file name to text area
        bytes.position = 18;
        compSize = bytes.readUnsignedInt();  // store size of compressed portion
        taFiles.text += "\tCompressed size is: " + compSize + '\n';
        bytes.position = 22;  // offset to uncompressed size
        uncompSize = bytes.readUnsignedInt();  // store uncompressed size
        taFiles.text += "\tUncompressed size is: " + uncompSize + '\n';
```

Im Beispiel wird der Rest der Datei für die durch die komprimierte Größe angegebene Länge aus dem Dateistrom in `bytes` gelesen und der Datei-Header in den ersten 30 Byte überschrieben. Die Angabe für die komprimierte Größe gilt auch dann, wenn die Datei nicht komprimiert wurde, da die komprimierte Größe in diesem Fall der entkomprimierten Größe der Datei entspricht.

```
    // read compressed file to offset 0 of bytes; for uncompressed files
    // the compressed and uncompressed size is the same
    if (compSize == 0) continue;
    zStream.readBytes(bytes, 0, compSize);
```

Im Beispiel wird dann die komprimierte Datei entkomprimiert und es wird die Funktion `outfile()` aufgerufen, um sie in den Ausgabedatenstrom zu schreiben. `outfile()`, der Dateiname und das Byte-Array, das die Dateidaten enthält, werden übergeben.

```
        if (compMethod == 8) // if file is compressed, uncompress
        {
            bytes.uncompress(CompressionAlgorithm.DEFLATE);
        }
        outFile(fileName, bytes);   // call outFile() to write out the file
```

Im vorigen Beispiel funktioniert `bytes.uncompress(CompressionAlgorithm.DEFLATE)` nur in AIR-Anwendungen. Um mit „deflate“ komprimierte Daten für AIR und Flash Player zu dekomprimieren, rufen Sie die `inflate()`-Funktion von ByteArray auf.

Die geschlossenen Klammern weisen auf das Ende der `while`-Schleife, der `init()`-Methode und des Flex-Anwendungscodes hin, mit Ausnahme der `outFile()`-Methode. Die Ausführung kehrt zum Beginn der Schleife `while` zurück und fährt mit der Verarbeitung der nächsten Byte in der .zip-Datei fort. Es wird eine weitere Datei extrahiert oder die Verarbeitung der .zip-Datei wird nach Verarbeitung der letzten Datei abgeschlossen.

```
    } // end of while loop
} // for Flex version, end of init() method and application
```

Die Funktion `outfile()` öffnet eine Ausgabedatei im Schreibmodus auf dem Desktop und versieht sie mit dem durch den Parameter `filename` angegebenen Namen. Dann schreibt die Funktion die Dateidaten aus dem Parameter `data` in den Ausgabedatenstrom (`outStream`) und schließt die Datei.

```
// Flash version
function outFile(fileName:String, data:ByteArray):void
{
    var outFile:File = File.desktopDirectory; // destination folder is desktop
    outFile = outFile.resolvePath(fileName);  // name of file to write
    var outStream:FileStream = new FileStream();
    // open output file stream in WRITE mode
    outStream.open(outFile, FileMode.WRITE);
    // write out the file
    outStream.writeBytes(data, 0, data.length);
    // close it
    outStream.close();
}

private function outFile(fileName:String, data:ByteArray):void
{
    var outFile:File = File.desktopDirectory; // dest folder is desktop
    outFile = outFile.resolvePath(fileName);  // name of file to write
    var outStream:FileStream = new FileStream();
    // open output file stream in WRITE mode
    outStream.open(outFile, FileMode.WRITE);
    // write out the file
    outStream.writeBytes(data, 0, data.length);
    // close it
    outStream.close();
}
```

Kapitel 42: Grundlagen zu Netzwerken und Kommunikation

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei der Erstellung von Anwendungen in Flash Player oder AIR müssen Sie häufig auf Ressourcen zugreifen, die sich außerhalb der Anwendung befinden. Beispielsweise senden Sie eine Bildanforderung an einen Webserver im Internet und erhalten die Bilddaten als Rückgabe. Oder Sie senden serialisierte Objekte über eine Socketverbindung mit einem Anwendungsserver hin und zurück. Die APIs für Flash Player und AIR bieten mehrere Klassen, über die Ihre Anwendungen an diesem Informationsaustausch teilnehmen können. Diese APIs unterstützen IP-Netzwerke mit Protokollen wie UDP, TCP, HTTP, RTMP und RTMFP.

Mit den folgenden Klassen können Daten über ein Netzwerk gesendet und empfangen werden:

Klasse	Unterstützte Datenformate	Protokolle	Beschreibung
Loader	SWF, PNG, JPEG, GIF	HTTP, HTTPS	Lädt unterstützte Datentypen und konvertiert die Daten in ein Anzeigeobjekt. Siehe „Dynamisches Laden von Anzeigeinhalten“ auf Seite 211.
URLLoader	Alle (Text, XML, binär usw.)	HTTP, HTTPS	Lädt beliebige Datenformate. Die Daten müssen von Ihrer Anwendung interpretiert werden. Siehe „Verwenden der URLLoader-Klasse“ auf Seite 869.
FileReference	Alle	HTTP	Dient zum Hoch- und Herunterladen von Dateien. Siehe „Verwenden der FileReference-Klasse“ auf Seite 695.
NetConnection	Video, Audio, ActionScript Message Format (AMF)	HTTP, HTTPS, RTMP, RTMFP	Stellt Verbindungen mit Video-, Audio- und Remote-Objektstreams her. Siehe „Verwenden von Videos“ auf Seite 505.
Sound	Audio	HTTP	Dient zum Laden und Abspielen von unterstützten Audioformaten. Siehe „Laden von externen Sounddateien“ auf Seite 474.
XMLSocket	XML	TCP	Tauscht XML-Nachrichten mit einem XMLSocket-Server aus. Siehe „XML-Sockets“ auf Seite 855.
Socket	Alle	TCP	Stellt eine Verbindung mit einem TCP-Socket-Server her. Siehe „Binäre Clientsockets“ auf Seite 850.
SecureSocket (AIR)	Alle	TCP mit SSL V3 oder TLS V1	Stellt eine Verbindung mit einem TCP-Socket-Server her, der SSL- oder TLS-Sicherheit erfordert. Siehe „Sichere Clientsockets (AIR)“ auf Seite 851.
ServerSocket (AIR)	Alle	TCP	Dient als Server für eingehende TCP-Socketverbindungen. Siehe „Serversockets“ auf Seite 859.

Klasse	Unterstützte Datenformate	Protokolle	Beschreibung
DatagramSocket (AIR)	Alle	UDP	Sendet und empfängt UDP-Pakete. Siehe „UDP-Sockets (AIR)“ auf Seite 861.

Bei der Erstellung einer Webanwendung ist es häufig ratsam, persistente Informationen über den Anwendungsstatus des Benutzers zu speichern. HTML-Seiten und -Anwendungen verwenden dazu normalerweise Cookies. In Flash Player können Sie zu diesem Zweck auch die SharedObject-Klasse verwenden. Siehe „Gemeinsame Objekte“ auf Seite 747. (Die SharedObject-Klasse kann in AIR-Anwendungen verwendet werden, es gelten jedoch weniger Einschränkungen, wenn die Daten einfach in einer regulären Datei gespeichert werden.)

Wenn Ihre Flash Player- oder AIR-Anwendung mit einer anderen Flash Player- oder AIR-Anwendung auf demselben Computer kommunizieren muss, können Sie die LocalConnection-Klasse verwenden. Beispielsweise können zwei (oder mehr) SWF-Dateien auf derselben Webseite miteinander kommunizieren. Genauso kann auch eine SWF-Datei, die in einer Webseite ausgeführt wird, mit einer AIR-Anwendung kommunizieren. Siehe „Kommunikation mit anderen Flash Player- und AIR-Instanzen“ auf Seite 884.

Zur Kommunikation mit anderen Prozessen (nicht SWF) auf dem lokalen Computer können Sie die NativeProcess-Klasse verwenden, die ab AIR 2 zur Verfügung steht. Über die NativeProcess-Klasse können Ihre AIR-Anwendungen andere Anwendungen starten und mit ihnen kommunizieren. Siehe „Kommunikation mit nativen Prozessen in AIR“ auf Seite 891.

Wenn Sie Informationen über die Netzwerkumgebung des Computers benötigen, auf dem eine AIR-Anwendung ausgeführt wird, können Sie die folgenden Klassen verwenden:

- NetworkInfo – Liefert Informationen über die verfügbaren Netzwerkschnittstellen, wie beispielsweise die IP-Adresse des Computers. Siehe „Netzwerkschnittstellen“ auf Seite 842.
- DNSResolver – Ermöglicht Ihnen die Suche nach DNS-Datensätzen. Siehe „DNS-Datensätze“ auf Seite 846.
- ServiceMonitor – Hiermit können Sie die Verfügbarkeit eines Servers überwachen. Siehe „Dienstüberwachung“ auf Seite 844.
- URLMonitor – Hiermit können Sie die Verfügbarkeit einer Ressource an einer bestimmten URL überwachen. Siehe „HTTP-Überwachung“ auf Seite 845.
- SocketMonitor und SecureSocketMonitor – Hiermit können Sie die Verfügbarkeit einer Ressource an einem Socket überwachen. Siehe „Socketüberwachung“ auf Seite 846.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die Ihnen beim Programmieren von Netzwerk- und Kommunikationscode begegnen:

Externe Daten Daten, die in irgendeiner Form außerhalb der Anwendung gespeichert sind und bei Bedarf in die Anwendung geladen werden. Diese Daten können in einer Datei gespeichert werden, die direkt geladen wird, oder in einer Datenbank oder in einer anderen Form, die durch Aufrufen von Skripts oder das Ausführen von Programmen auf einem Server abgerufen wird.

URL-kodierte Variablen Im URL-kodierten Format können mehrere Variablen (Paare aus Variablennamen und -werten) in einem Textstring dargestellt werden. Einzelne Variablen werden im Format „Name=Wert“ geschrieben. Die Variablen (d. h. die Name-Wert-Paare) sind durch Und-Zeichen voneinander getrennt: Variable1=Wert1&Variable2=Wert2. Auf diese Weise kann eine unbegrenzte Anzahl von Variablen in einer Nachricht gesendet werden.

MIME-Typ Ein Standardcode, mit dem der Typ einer Datei bei der Internet-Kommunikation identifiziert wird. Jeder Dateityp weist einen bestimmten Code auf, der zu seiner Identifikation dient. Beim Senden einer Datei oder einer

Nachricht gibt ein Computer (z. B. ein Webserver oder die Flash Player- oder AIR-Instanz eines Benutzers) den Typ der gesendeten Datei an.

HTTP Hypertext Transfer Protocol - ein Standardformat für die Bereitstellung von Webseiten und verschiedenen anderen Inhaltstypen, die über das Internet gesendet werden.

Anforderungsmethode Wenn eine Anwendung (zum Beispiel eine AIR-Anwendung oder ein Webbrowser) eine Nachricht (eine sogenannte HTTP-Anforderung) an einen Webserver sendet, können alle mit dieser Anforderung gesendeten Daten auf eine von zwei Arten eingebettet werden. Entsprechend gibt es die beiden Anforderungsmethoden GET und POST. Auf dem Server muss das Programm, das die Anforderung empfängt, den entsprechenden Teil der Anforderung untersuchen, um die Daten zu finden. Daher muss die zum Senden der Daten aus Ihrer Anwendung verwendete Anforderungsmethode der Anforderungsmethode entsprechen, die auf dem Server zum Lesen der Daten verwendet wird.

Socketverbindung Eine permanente Kommunikationsverbindung zwischen zwei Computern.

Hochladen Das Senden einer Datei an einen anderen Computer.

Herunterladen Das Empfangen einer Datei von einem anderen Computer.

Netzwerkschnittstellen

Adobe AIR 2 und höher

Mithilfe des NetworkInfo-Objekts können Sie die Hardware- und Softwareschnittstellen ermitteln, die für Ihre Anwendung zur Verfügung stehen. Das NetworkInfo-Objekt ist ein *singleton*-Objekt; Sie müssen kein solches Objekt erstellen. Verwenden Sie stattdessen die statische Klasseneigenschaft `networkInfo`, um auf das einzelne NetworkInfo-Objekt zuzugreifen. Das NetworkInfo-Objekt löst auch ein `networkChange`-Ereignis aus, wenn eine der verfügbaren Schnittstellen sich ändert.

Rufen Sie die `findInterfaces()`-Methode auf, um eine Liste der NetworkInterface-Objekte abzurufen. Jedes NetworkInterface-Objekt in der Liste beschreibt eine der verfügbaren Schnittstellen. Das NetworkInterface-Objekt bietet Informationen wie die IP-Adresse, Hardwareadresse und maximale Übertragungseinheit. Weiterhin gibt es an, ob die Schnittstelle aktiv ist.

Mit dem folgenden Codebeispiel werden die NetworkInterface-Eigenschaften jeder Schnittstelle auf dem Clientcomputer verfolgt:

```
package {
import flash.display.Sprite;
import flash.net.InterfaceAddress;
import flash.net.NetworkInfo;
import flash.net.NetworkInterface;

public class NetworkInformationExample extends Sprite
{
    public function NetworkInformationExample()
    {
        var networkInfo:NetworkInfo = NetworkInfo.networkInfo;
        var interfaces:Vector.<NetworkInterface> = networkInfo.findInterfaces();

        if( interfaces != null )
        {
            trace( "Interface count: " + interfaces.length );
            for each ( var interfaceObj:NetworkInterface in interfaces )
            {
                trace( "\nname: " + interfaceObj.name );
                trace( "display name: " + interfaceObj.displayName );
                trace( "mtu: "  + interfaceObj.mtu );
                trace( "active?: " + interfaceObj.active );
                trace( "parent interface: " + interfaceObj.parent );
                trace( "hardware address: " + interfaceObj.hardwareAddress );
                if( interfaceObj.subInterfaces != null )
                {
                    trace( "# subinterfaces: " + interfaceObj.subInterfaces.length );
                }
                trace("# addresses: " + interfaceObj.addresses.length );
                for each ( var address:InterfaceAddress in interfaceObj.addresses )
                {
                    trace( "  type: "           + address.ipVersion );
                    trace( "  address: " + address.address );
                    trace( "  broadcast: " + address.broadcast );
                    trace( "  prefix length: " + address.prefixLength );
                }
            }
        }
    }
}
}
```

Weitere Informationen finden Sie unter:

- NetworkInfo
- NetworkInterface
- InterfaceAddress
- Flexpert: Detecting the network connection type with Flex 4.5

Änderungen an Netzwerkverbindungen

Adobe AIR 1.0 und höher

Die AIR-Anwendung kann in Umgebungen mit unsicheren und sich ändernden Netzwerkverbindungen ausgeführt werden. Adobe AIR unterstützt eine Anwendung bei der Verwaltung von Verbindungen zu Onlineressourcen durch Senden eines Netzwerkänderungsereignisses, wenn der Status einer Netzwerkverbindung zu verfügbar oder nicht verfügbar wechselt. Sowohl das NetworkInfo-Objekt als auch das NativeApplication-Objekt der Anwendung lösen das `networkChange`-Ereignis aus. Fügen Sie einen Listener hinzu, um auf dieses Ereignis reagieren zu können:

```
NetworkInfo.networkInfo.addEventListener(Event.NETWORK_CHANGE, onNetworkChange);
```

Definieren Sie ferner eine Ereignisprozedurfunktion:

```
function onNetworkChange(event:Event)
{
    //Check resource availability
}
```

Das `networkChange`-Ereignis weist nicht auf eine Änderung der gesamten Netzwerkaktivität hin, sondern nur darauf, dass sich eine einzelne Netzwerkverbindung geändert hat. AIR unternimmt keinen Versuch, die Bedeutung der Netzwerkänderung zu interpretieren. Ein vernetzter Computer kann über zahlreiche reale und virtuelle Verbindungen verfügen. Der Verlust einer Verbindung führt nicht zwangsläufig zum Verlust einer Ressource. Andererseits sind neue Verbindungen auch keine Garantie für eine bessere Ressourcenverfügbarkeit. Manchmal kann eine neue Verbindung sogar den Zugriff auf zuvor verfügbare Ressourcen blockieren (z. B. beim Zugreifen auf ein VPN).

Im Allgemeinen kann eine Anwendung nur herausfinden, ob eine Verbindung zu einer Remoteressource hergestellt werden kann, indem sie versucht, die Verbindung herzustellen. Die Dienstüberwachungsarchitektur bietet eine ereignisgestützte Methode, auf Änderungen der Netzwerkverbindung mit einem bestimmten Host zu reagieren.

Hinweis: *Die Dienstüberwachungsarchitektur erkennt, ob ein Server auf akzeptable Weise auf eine Anforderung reagiert. Eine erfolgreiche Prüfung gewährleistet keine vollständige Konnektivität. Skalierbare Webdienste verwenden häufig Applicances mit Cachefunktion oder Lastausgleich, um Datenverkehr an einen Webservercluster umzuleiten. In diesem Fall bieten Dienstanbieter nur eine teilweise Diagnose der Netzwerkverbindung.*

Dienstüberwachung

Adobe AIR 1.0 und höher

Die von der AIR-Architektur separate Dienstüberwachungsarchitektur befindet sich in der Datei „aircore.swc“. Die Architektur kann nur verwendet werden, wenn der Generierungsprozess die Datei „aircore.swc“ umfasst.

Bei Adobe® Flash® Builder ist diese Bibliothek automatisch enthalten.

Die ServiceMonitor-Klasse implementiert die Architektur zum Überwachen von Netzwerkdiensten und bietet grundlegende Funktionen für die Dienstüberwachung. Eine Instanz der ServiceMonitor-Klasse löst standardmäßig Ereignisse in Bezug auf die Netzwerkverbindung aus. Das ServiceMonitor-Objekt löst diese Ereignisse aus, wenn die Instanz erstellt wird und wenn die Laufzeit eine Netzwerkänderung erkennt. Sie können darüber hinaus festlegen, dass die `pollInterval`-Eigenschaft einer ServiceMonitor-Instanz die Verbindung unabhängig von allgemeinen Netzwerkverbindungsereignissen in regelmäßigen Abständen in Millisekunden prüft. Ein ServiceMonitor-Objekt prüft die Netzwerkverbindung erst beim Aufrufen der `start()`-Methode.

Die URLMonitor-Klasse, eine Unterklasse der ServiceMonitor-Klasse, erkennt Änderungen bei der HTTP-Verbindung für ein bestimmtes URLRequest.

Die SocketMonitor-Klasse, ebenfalls eine Unterklasse der ServiceMonitor-Klasse, erkennt Änderungen bei der Verbindung mit einem bestimmten Host an einem bestimmten Port.

Hinweis: *Vor AIR 2 wurde die Dienstüberwachungsarchitektur in der Bibliothek „servicemonitor.swc" veröffentlicht. Diese Bibliothek ist nun veraltet. Verwenden Sie stattdessen die Bibliothek „aircore.swc".*

Flash CS4 und CS5 Professional

So verwenden Sie diese Klassen in Adobe® Flash® CS4 oder CS5 Professional:

1. Wählen Sie „Datei" > „Einstellungen für Veröffentlichungen".
2. Klicken Sie auf die Schaltfläche „Einstellungen" für ActionScript 3.0. Wählen Sie „Bibliothekspfad".
3. Klicken Sie auf „Nach SWC-Datei suchen" und navigieren Sie zum AIK-Ordner im Installationsordner von Flash Professional.
4. Suchen Sie in diesem Ordner nach /frameworks/libs/air/aircore.swc (für AIR 2) oder /frameworks/libs/air/servicemonitor.swc (für AIR 1.5).
5. Klicken Sie auf „OK".
6. Fügen Sie dem ActionScript 3.0-Code die folgende Import-Anweisung hinzu:

```
import air.net.*;
```

Flash CS3 Professional

Um diese Klassen in Adobe® Flash® CS3 Professional zu verwenden, ziehen Sie die ServiceMonitorShim-Komponente aus dem Bedienfeld „Komponenten" in das Bedienfeld „Bibliothek". Fügen Sie dann die folgende `import`-Anweisung in den ActionScript 3.0-Code ein:

```
import air.net.*;
```

HTTP-Überwachung

Adobe AIR 1.0 und höher

Die URLMonitor-Klasse ermittelt, ob HTTP-Anforderungen an einer bestimmten Adresse an Port 80 (dem regulären Port für HTTP-Kommunikation) gestellt werden können. Der folgende Code verwendet eine Instanz der URLMonitor-Klasse, um Verbindungsänderungen zur Adobe-Website zu erkennen:

```
import air.net.URLMonitor;
import flash.net.URLRequest;
import flash.events.StatusEvent;
var monitor:URLMonitor;
monitor = new URLMonitor(new URLRequest('http://www.example.com'));
monitor.addEventListener(StatusEvent.STATUS, announceStatus);
monitor.start();
function announceStatus(e:StatusEvent):void {
    trace("Status change. Current status: " + monitor.available);
}
```

Socketüberwachung

Adobe AIR 1.0 und höher

AIR-Anwendungen können auch Socketverbindungen für Push-Model-Verbindungen verwenden. Firewalls und Netzwerkrouter beschränken die Netzwerkkommunikation an nicht autorisierten Ports normalerweise aus Sicherheitsgründen. Deshalb müssen Entwickler in Erwägung ziehen, dass Benutzer nicht immer die Möglichkeit haben, Socketverbindungen herzustellen.

Der folgende Code verwendet eine Instanz der SocketMonitor-Klasse, um Änderungen an einer Socketverbindung zu erkennen. Der überwachte Port ist 6667, der häufig für IRC verwendet wird:

```
import air.net.ServiceMonitor;
import flash.events.StatusEvent;

socketMonitor = new SocketMonitor('www.example.com',6667);
socketMonitor.addEventListener(StatusEvent.STATUS, socketStatusChange);
socketMonitor.start();

function announceStatus(e:StatusEvent):void {
    trace("Status change. Current status: " + socketMonitor.available);
}
```

Wenn der Socket-Server eine sichere Verbindung erfordert, können Sie die SecureSocketMonitor-Klasse anstelle von SocketMonitor verwenden.

DNS-Datensätze

Adobe AIR 2.0 und höher

DNS-Ressourcendatensätze (DNS = Domain Name System) können mithilfe der DNSResolver-Klasse ermittelt werden. DNS-Ressourcendatensätze bieten Informationen wie die IP-Adresse eines Domänennamens und den Domänennamen einer IP-Adresse. Sie können folgende Arten von DNS-Ressourcendatensätzen ermitteln:

- ARecord – IPv4-Adresse für einen Host.
- AAAARecord – IPv6-Adresse für einen Host.
- MXRecord – E-Mail-Datensatz für einen Host.
- PTRRecord – Hostname für eine IP-Adresse.
- SRVRecord – Dienstdatensatz für einen Dienst..

Zum Ermitteln eines Datensatzes übergeben Sie einen Abfragestring und das Klassenobjekt für den jeweiligen Datensatztyp an die `lookup()`-Methode des DNSResolver-Objekts. Der verwendete Abfragestring richtet sich nach dem Typ des Datensatzes:

Record-Klasse	Abfragestring	Beispiel für Abfragestring
ARecord	Hostname	“example.com”
AAAARecord	Hostname	“example.com”
MXRecord	Hostname	“example.com”
PTRRecord	IP-Adresse	“208.77.188.166”
SRVRecord	Dienstbezeichner: _service._protocol.host	“_sip._tcp.example.com”

Mit dem folgenden Codebeispiel wird die IP-Adresse des Hosts „example.com“ ermittelt.

```
package
{
    import flash.display.Sprite;
    import flash.events.DNSResolverEvent;
    import flash.events.ErrorEvent;
    import flash.net.dns.ARecord;
    import flash.net.dns.DNSResolver;

    public class DNSResolverExample extends Sprite
    {

        public function DNSResolverExample()
        {
            var resolver:DNSResolver = new DNSResolver();
            resolver.addEventListener( DNSResolverEvent.LOOKUP, lookupComplete );
            resolver.addEventListener( ErrorEvent.ERROR, lookupError );

            resolver.lookup( "example.com.", ARecord );
        }

        private function lookupComplete( event:DNSResolverEvent ):void
        {
            trace( "Query string: " + event.host );
            trace( "Record count: " + event.resourceRecords.length );
            for each( var record:* in event.resourceRecords )
            {
                if( record is ARecord ) trace( record.address );
            }

        }

        private function lookupError( error:ErrorEvent ):void
        {
            trace("Error: " + error.text );
        }
    }
}
```

Weitere Informationen finden Sie unter:

- DNSResolver
- DNSResolverEvent

- ARecord
- AAAARecord
- MXRecord
- PTRRecord
- SRVRecord

Kapitel 43: Sockets

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Socket ist eine bestimmte Art von Netzwerkverbindung zwischen zwei Computerprozessen. Normalerweise werden die Prozesse auf zwei verschiedenen Computern ausgeführt, die mit demselben IP-Netzwerk verbunden sind. Die verbundenen Prozesse können jedoch auf demselben Computer ausgeführt werden, wobei die spezielle IP-Adresse für den „lokalen Host“ verwendet wird.

Adobe Flash Player unterstützt clientseitige TCP-Sockets (Transport Control Protocol). Eine Flash Player-Anwendung kann eine Verbindung mit einem anderen Prozess herstellen, der als Socket-Server agiert, aber keine eingehenden Verbindungsanforderungen von anderen Prozessen entgegennehmen. Anders ausgedrückt: Eine Flash Player-Anwendung kann eine Verbindung mit einem TCP-Server herstellen, aber nicht als TCP-Server fungieren.

Die Flash Player-API enthält auch die XMLSocket-Klasse. Die XMLSocket-Klasse verwendet ein spezifisches Flash Player-Protokoll, das den Austausch von XML-Nachrichten mit einem Server ermöglicht, der dieses Protokoll interpretieren kann. Die XMLSocket-Klasse wurde in ActionScript 1 eingeführt und wird zur Gewährleistung der Abwärtskompatibilität auch weiterhin unterstützt. Im Allgemeinen sollte für neue Anwendungen die Socket-Klasse verwendet werden, es sei denn, Sie stellen eine Verbindung mit einem Server her, der konkret für die Kommunikation mit Flash XMLSockets eingerichtet wurde.

In Adobe AIR stehen mehrere zusätzliche Klassen für die socketbasierte Netzwerkprogrammierung zur Verfügung. AIR-Anwendungen können mit der ServerSocket-Klasse als TCP-Socket-Server agieren und über die SecureSocket-Klasse Verbindungen mit Socket-Servern herstellen, die SSL- oder TLS-Sicherheit erfordern. AIR-Anwendungen können auch UDP-Nachrichten (Universal Datagram Protocol) über die DatagramSocket-Klasse senden und empfangen.

Verwandte Hilfethemen

flash.net-Paket

„Herstellen einer Verbindung mit Sockets“ auf Seite 1145

TCP-Sockets

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das TCP-Protokoll (Transmission Control Protocol) bietet die Möglichkeit, Nachrichten über eine dauerhafte Netzwerkverbindung auszutauschen. TCP gewährleistet, dass die gesendeten Nachrichten in der richtigen Reihenfolge eingehen (sofern keine größeren Netzwerkprobleme vorliegen). TCP-Verbindungen erfordern einen Client und einen Server. Flash Player kann Client-Sockets erstellen. Adobe AIR kann zudem Server-Sockets erstellen.

Die folgenden ActionScript-APIs ermöglichen TCP-Verbindungen:

- Socket – ermöglicht einer Client-Anwendung, eine Verbindung mit einem Server herzustellen. Die Socket-Klasse kann nicht auf eingehende Verbindungen warten.
- SecureSocket (AIR) – ermöglicht einer Client-Anwendung, eine Verbindung mit einem vertrauenswürdigen Server herzustellen und eine verschlüsselte Kommunikation durchzuführen.

- ServerSocket (AIR) – ermöglicht einer Anwendung, auf eingehende Verbindungen zu warten und als Server zu agieren.
- XMLSocket – ermöglicht einer Client-Anwendung, eine Verbindung mit einem XMLSocket-Server herzustellen.

Binäre Clientsockets

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine binäre Socketverbindung ähnelt einem XML-Socket, außer dass Client und Server nicht auf den Austausch von XML-Nachrichten beschränkt sind. Stattdessen kann die Verbindung Daten als binäre Informationen übertragen. Dadurch können Verbindungen mit zahlreichen Diensten hergestellt werden, einschließlich Mailserver (POP3, SMTP und IMAP) und News-Server (NNTP).

Socket-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Socket-Klasse ermöglicht die Herstellung von Socketverbindungen sowie das Lesen und Schreiben von unformatierten Binärdaten. Die Socket-Klasse eignet sich gut für die Arbeit mit Servern, die Binärprotokolle verwenden. Durch die Verwendung von binären Socketverbindungen können Sie Code schreiben, der mit mehreren verschiedenen Internetprotokollen interagiert, wie POP3, SMTP, IMAP und NNTP. Diese Interaktion ermöglicht es Ihren Anwendungen, eine Verbindung zu Mailservern und News-Servern herzustellen.

Flash Player kann bei der Datenübertragung mit einem Server direkt das Binärprotokoll dieses Servers verwenden. Einige Server verwenden die Byte-Reihenfolge „big-endian“ und andere die Byte-Reihenfolge „little-endian“. Die meisten Server im Internet verwenden die Byte-Reihenfolge „big-endian“, da dies der Byte-Reihenfolge in Netzwerken entspricht. Die Bytereihenfolge littleEndian ist verbreitet, da sie in der Intel® x86-Prozessorarchitektur verwendet wird. Sie sollten die Byte-Reihenfolge verwenden, die der Byte-Reihenfolge des Servers entspricht, der die Daten sendet oder empfängt. Alle Vorgänge, die von den Schnittstellen „IDataInput“ und „IDataOutput“ ausgeführt werden, und die Klassen, die diese Schnittstellen implementieren (ByteArray, Socket und URLStream), liegen standardmäßig im bigEndian-Format vor, dabei steht das höchstwertige Byte an erster Stelle. Diese standardmäßige Byte-Reihenfolge wurde wegen der Übereinstimmung mit Java und der offiziellen Byte-Reihenfolge für Netzwerke (Network Byte Order) gewählt. Um von der bigEndian- zur littleEndian-Bytereihenfolge oder umgekehrt zu wechseln, stellen Sie die `endian`-Eigenschaft auf `Endian.BIG_ENDIAN` oder `Endian.LITTLE_ENDIAN` ein.

Die Socket-Klasse übernimmt alle Methoden, die von den IDataInput- und IDataOutput-Schnittstellen definiert werden (im flash.utils-Paket). Diese Methoden müssen für Lese- und Schreibvorgänge mit den Sockets verwendet werden.

Weitere Informationen finden Sie unter:

- Socket
- IDataInput
- IDataOutput
- socketData-Ereignis

Sichere Clientsockets (AIR)

Adobe AIR 2 und höher

Mithilfe der SecureSocket-Klasse können Sie Verbindungen mit Socketservern herstellen, die SSL-Version 4 (Secure Sockets Layer) oder TLS-Version 1 (Transport Layer Security) verwenden. Sichere Sockets bieten drei Vorteile: Serverauthentifizierung, Datenintegrität und Vertraulichkeit der Nachrichten. Die Laufzeit authentifiziert einen Server anhand des Serverzertifikats und der Beziehung des Servers mit den Stamm- oder Zwischenzertifikaten der entsprechenden Zertifizierungsstellen im Vertrauensspeicher des Benutzers. Zur Gewährleistung von Datenintegrität und Vertraulichkeit der Nachrichten stützt sich die Laufzeit auf die Verschlüsselungsalgorithmen der SSL- und TLS-Protokollimplementierungen.

Wenn Sie über das SecureSocket-Objekt eine Verbindung mit einem Server herstellen, validiert die Laufzeit das Serverzertifikat anhand des Zertifikatvertrauensspeichers. Unter Windows und Mac stellt das Betriebssystem den Vertrauensspeicher bereit. Unter Linux bietet die Laufzeitumgebung einen eigenen Vertrauensspeicher.

Wenn das Serverzertifikat nicht gültig oder vertrauenswürdig ist, gibt die Laufzeit ein `ioError`-Ereignis aus. Anhand der `serverCertificateStatus`-Eigenschaft des SecureSocket-Objekts können Sie überprüfen, warum die Validierung fehlgeschlagen ist. Für Server ohne gültiges und vertrauenswürdiges Zertifikat steht kein Kommunikationsmechanismus zur Verfügung.

Die CertificateStatus-Klasse definiert Stringkonstanten für die möglichen Validierungsergebnisse:

- Expired – Das Zertifikat ist abgelaufen.
- Invalid – Das Zertifikat ist ungültig. Dies kann mehrere Gründe haben. Beispielsweise wurde das Zertifikat geändert oder beschädigt oder es hat nicht den richtigen Typ.
- Invalid chain – Mindestens ein Zertifikat in der Zertifikatkette des Servers ist ungültig.
- Principal mismatch – Der Hostname des Servers und der Zertifikatname stimmen nicht überein. Der Server verwendet also das falsche Zertifikat.
- Revoked – Die ausgebende Zertifizierungsstelle hat das Zertifikat gesperrt.
- Trusted – Das Zertifikat ist gültig und vertrauenswürdig. Ein SecureSocket-Objekt kann nur eine Verbindung mit einem Server herstellen, der ein gültiges, vertrauenswürdiges Zertifikat verwendet.
- Unknown – Das SecureSocket-Objekt hat das Zertifikat noch nicht validiert. Die `serverCertificateStatus`-Eigenschaft hat diesen Statuswert, bevor Sie `connect()` aufrufen und bevor entweder ein `connect`- oder ein `ioError`-Ereignis ausgelöst wird.
- Untrusted signers – Das Zertifikat ist nicht mit einem vertrauenswürdigen Stammzertifikat im Vertrauensspeicher des Clientcomputers „verkettet“.

Für die Kommunikation mit einem SecureSocket-Objekt muss der Server ein sicheres Protokoll verwenden und über ein gültiges und vertrauenswürdiges Zertifikat verfügen. In jeder anderen Hinsicht entspricht die Verwendung eines SecureSocket-Objekts der Verwendung eines Socket-Objekts.

Das SecureSocket-Objekt wird nicht auf allen Plattformen unterstützt. Verwenden Sie die `isSupported`-Eigenschaft der SecureSocket-Klasse, um zu überprüfen, ob die Laufzeitumgebung die Verwendung des SecureSocket-Objekts auf dem aktuellen Clientcomputer unterstützt.

Weitere Informationen finden Sie unter:

- SecureSocket
- CertificateStatus
- IDataInput

- IDataOutput
- socketData-Ereignis

TCP-Socket-Beispiel: Erstellen eines Telnet-Clients

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Telnet-Beispiel demonstriert Techniken für das Herstellen einer Verbindung mit einem Remote-Server und das Übertragen von Daten mithilfe der Socket-Klasse. In diesem Beispiel werden die folgenden Verfahren veranschaulicht:

- Erstellen eines benutzerdefinierten Telnet-Clients mithilfe der Socket-Klasse
- Senden von Text an den Remote-Server mithilfe eines ByteArray-Objekts
- Verarbeiten der von einem Remote-Server empfangenen Daten

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Telnet-Anwendung finden Sie im Ordner „Samples/Telnet“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
TelnetSocket.fla oder TelnetSocket.mxml	Die Hauptanwendungsdatei mit der Benutzeroberfläche für Flex (MXML) oder Flash (FLA).
TelnetSocket.as	Document-Klasse, die die Benutzeroberflächenlogik bereitstellt (nur Flash).
com/example/programmingas3/Telnet/Telnet.as	Stellt die Telnet-Clientfunktionsmerkmale für die Anwendung bereit. Hierzu gehören das Herstellen einer Verbindung mit einem Remote-Server sowie das Senden, Empfangen und Anzeigen von Daten.

Überblick über die Anwendung „TelnetSocket“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Hauptdatei „TelnetSocket.mxml“ wird die Benutzeroberfläche (User Interface, UI) der gesamten Anwendung erstellt.

Zusätzlich zur Benutzeroberfläche sind in dieser Datei die beiden Methoden `login()` und `sendCommand()` definiert, mit denen eine Verbindung zwischen Benutzer und angegebenem Server hergestellt wird.

Im Folgenden ist der ActionScript-Code der Hauptanwendungsdatei aufgeführt:

```
import com.example.programmingas3.socket.Telnet;

private var telnetClient:Telnet;
private function connect():void
{
    telnetClient = new Telnet(serverName.text, int(portNumber.text), output);
    console.title = "Connecting to " + serverName.text + ":" + portNumber.text;
    console.enabled = true;
}
private function sendCommand():void
{
    var ba:ByteArray = new ByteArray();
    ba.writeMultiByte(command.text + "\n", "UTF-8");
    telnetClient.writeBytesToSocket(ba);
    command.text = "";
}
```

Mit der ersten Codezeile wird die Telnet-Klasse aus dem benutzerdefinierten com.example.programmingas.socket-Paket importiert. Mit der zweiten Codezeile wird eine Instanz der Telnet-Klasse deklariert, `telnetClient`, die später von der `connect()`-Methode initialisiert wird. Dann wird die `connect()`-Methode deklariert und die zuvor deklarierte Variable `telnetClient` initialisiert. Diese Methode übergibt den benutzerdefinierten Telnet-Servernamen „telnet server port“ und einen Verweis auf die TextArea-Komponente an die Anzeigeliste, die zur Anzeige der Textantworten vom Socketserver verwendet wird. Die letzten beiden Zeilen der `connect()`-Methode stellen die `title`-Eigenschaft für das Panel ein und aktivieren die Panel-Komponente, die es dem Benutzer ermöglicht, Daten an den Remote-Server zu senden. Die letzte Methode in der Hauptanwendungsdatei, `sendCommand()`, dient zum Senden der Benutzerbefehle als ein ByteArray-Objekt an den Remote-Server.

Überblick über die Telnet-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Telnet-Klasse ist für das Herstellen einer Verbindung mit dem externen Telnet-Server und das Senden/Empfangen von Daten verantwortlich.

Die Telnet-Klasse deklariert die folgenden privaten Variablen:

```
private var serverURL:String;
private var portNumber:int;
private var socket:Socket;
private var ta:TextArea;
private var state:int = 0;
```

Die erste Variable, `serverURL`, enthält die vom Benutzer eingegebene Serveradresse, mit der eine Verbindung hergestellt werden soll.

Die zweite Variable `portNumber` ist die Nummer des Anschlusses, auf dem der Telnet-Server derzeit ausgeführt wird. In der Standardeinstellung befindet sich der Telnet-Dienst auf dem Anschluss 23.

Die dritte Variable `socket` ist eine Socket-Instanz, die versucht, eine Verbindung mit dem von den Variablen `serverURL` und `portNumber` angegebenen Server herzustellen.

Die vierte Variable `ta` ist ein Verweis auf eine Instanz der TextArea-Komponente auf der Bühne. Diese Komponente dient zum Anzeigen der Antworten vom externen Telnet-Server sowie aller möglicher Fehlermeldungen.

Die letzte Variable, `state`, ist ein numerischer Wert, mit dem festgestellt wird, welche Optionen Ihr Telnet-Client unterstützt.

Wie Sie bereits gesehen haben, wird die Konstruktorfunktion der Telnet-Klasse von der `connect()`-Methode in der Hauptanwendungsdatei aufgerufen.

Der Telnet-Konstruktor lässt drei Parameter zu: `server`, `port` und `output`. Die Parameter `server` und `port` geben den Servernamen und die Anschlussnummer an, auf welcher der Telnet-Server ausgeführt wird. Der letzte Parameter, `output`, ist ein Verweis auf die Instanz einer TextArea-Komponente auf der Bühne, in der die Serverausgabe angezeigt wird.

```
public function Telnet(server:String, port:int, output:TextArea)
{
    serverURL = server;
    portNumber = port;
    ta = output;
    socket = new Socket();
    socket.addEventListener(Event.CONNECT, connectHandler);
    socket.addEventListener(Event.CLOSE, closeHandler);
    socket.addEventListener(ErrorEvent.ERROR, errorHandler);
    socket.addEventListener(IOErrorEvent.IO_ERROR, ioErrorHandler);
    socket.addEventListener(ProgressEvent.SOCKET_DATA, dataHandler);
    Security.loadPolicyFile("http://" + serverURL + "/crossdomain.xml");
    try
    {
        msg("Trying to connect to " + serverURL + ":" + portNumber + "\n");
        socket.connect(serverURL, portNumber);
    }
    catch (error:Error)
    {
        msg(error.message + "\n");
        socket.close();
    }
}
```

Schreiben von Daten an eine Socketverbindung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Schreiben von Daten an eine Socketverbindung rufen Sie eine der Schreibmethoden (write) der Socket-Klasse auf. Zu diesen Schreibmethoden zählen `writeBoolean()`, `writeByte()`, `writeBytes()` und `writeDouble()`. Dann löschen Sie die Daten mit der `flush()`-Methode aus dem Ausgabepuffer. Im Telnet-Server werden Daten mithilfe der `writeBytes()`-Methode an die Socketverbindung geschrieben. Die Methode verarbeitet das Byte-Array als einen Parameter und sendet ihn an den Ausgabepuffer. Die `writeBytesToSocket()`-Methode lautet folgendermaßen:

```
public function writeBytesToSocket(ba:ByteArray):void
{
    socket.writeBytes(ba);
    socket.flush();
}
```

Diese Methode wird von der `sendCommand()`-Methode der Hauptanwendungsdatei aufgerufen.

Anzeigen von Nachrichten vom Socketserver

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Immer wenn eine Nachricht vom Socketserver empfangen wird oder ein Ereignis auftritt, wird die benutzerdefinierte `msg()`-Methode aufgerufen. Diese Methode fügt einen String an die TextArea-Komponente auf der Bühne an und ruft eine benutzerdefinierte `setScroll()`-Methode auf, die dafür sorgt, dass in der TextArea-Komponente ein Bildlauf bis zum Ende durchgeführt wird. Die `msg()`-Methode lautet folgendermaßen:

```
private function msg(value:String):void
{
    ta.text += value;
    setScroll();
}
```

Wenn Sie den Inhalt der TextArea-Komponente nicht automatisch scrollen, müssten die Benutzer die Bildlaufleisten im Textbereich manuell ziehen, um die letzte Antwort vom Server lesen zu können.

Durchführen eines Bildlaufs in einer TextArea-Komponente

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `setScroll()`-Methode enthält eine einzelne ActionScript-Codezeile, mit der für den Inhalt der TextArea-Komponente ein vertikaler Bildlauf durchgeführt wird, sodass der Benutzer die letzte Zeile des zurückgegebenen Textes lesen kann. Die Methode `setScroll()` ist im folgenden Codeausschnitt dargestellt:

```
public function setScroll():void
{
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}
```

Diese Methode stellt die Eigenschaft `verticalScrollPosition` ein, bei der es sich um die Zeilennummer der ersten angezeigten Zeichenreihe handelt, und stellt sie auf den Wert der Eigenschaft `maxVerticalScrollPosition` ein.

XML-Sockets

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit XML-Sockets können Sie eine Verbindung mit einem Remoteserver erstellen. Die Verbindung bleibt so lange geöffnet, bis sie explizit geschlossen wird. Sie können Stringdaten, wie XML, zwischen Server und Client austauschen. Ein XML-Socketserver bietet den Vorteil, dass der Client die Daten nicht ausdrücklich anzufordern braucht. Der Server kann die Daten senden, ohne auf eine Anforderung warten zu müssen. Die Daten können an alle angeschlossenen Clients gesendet werden.

Für Flash Player und für Adobe AIR-Inhalt außerhalb der Anwendungs-Sandbox muss für XML-Socketverbindungen eine Socket-Richtliniendatei auf dem Zielserver vorhanden sein. Weitere Informationen finden Sie unter „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125 und „Herstellen einer Verbindung mit Sockets“ auf Seite 1145.

Die XMLSocket-Klasse kann nicht automatisch einen Tunnel durch Firewalls öffnen, da XMLSocket im Gegensatz zum RTMP-Protokoll (Real-Time Messaging Protocol) nicht über HTTP-Tunnelfähigkeiten verfügt. Wenn Sie HTTP-Tunneling verwenden müssen, sollten Sie die Verwendung von Flash Remoting oder Flash Mail Server in Betracht ziehen, da diese RTMP unterstützen.

Folgende Einschränkungen gelten bei Inhalt in Flash Player oder in einer AIR-Anwendung außerhalb der Sicherheits-Sandbox der Anwendung für die Herstellung einer Verbindung zwischen einem XMLSocket-Objekt und dem Server:

- Bei Inhalt außerhalb der Sicherheits-Sandbox der Anwendung können mit der `XMLSocket.connect()`-Methode nur Verbindungen zu TCP-Portnummern größer oder gleich 1024 hergestellt werden. Als Folge dieser Einschränkung müssen den Serverdaemonprogrammen, die mit dem `XMLSocket`-Objekt kommunizieren, ebenfalls Portnummern größer oder gleich 1024 zugeordnet sein. Portnummern unter 1024 werden häufig von Systemdiensten wie FTP (21), Telnet (23), SMTP (25), HTTP (80) und POP3 (110) verwendet, und die Verwendung dieser Anschlüsse durch XMLSocket-Objekte ist aus Sicherheitsgründen nicht zulässig. Durch Einschränkung der verfügbaren Portnummern lässt sich das Risiko verringern, dass in ungeeigneter oder missbräuchlicher Weise auf diese Ressourcen zugegriffen wird.
- Bei Inhalt außerhalb der Sicherheits-Sandbox der Anwendung können mit der `XMLSocket.connect()`-Methode nur Verbindungen zu Computern in derselben Domäne, in der sich der Inhalt befindet, hergestellt werden. (Diese Einschränkung ist identisch mit den Sicherheitsregeln für `URLLoader.load()`.) Wenn Sie eine Verbindung zu einem Serverdaemon herstellen möchten, der sich in einer anderen Domäne als der des Inhalts befindet, können Sie eine domänenübergreifende Richtliniendatei auf dem Server erstellen, mit dem der Zugriff von bestimmten Domänen möglich ist. Weitere Informationen zu domänenübergreifenden Richtliniendateien finden Sie unter „AIR-Sicherheit“ auf Seite 1153.

Hinweis: *Die Einrichtung eines Servers zur Kommunikation mit einem XMLSocket-Objekt ist oft recht anspruchsvoll. Wenn Ihre Anwendung keine Interaktionen in Echtzeit erfordert, verwenden Sie die URLLoader-Klasse anstelle der XMLSocket-Klasse.*

Mit den Methoden `XMLSocket.connect()` und `XMLSocket.send()` der XMLSocket-Klasse können Sie XML-Daten über eine Socketverbindung mit einem Server austauschen. Mit der Methode `XMLSocket.connect()` stellen Sie eine Socketverbindung zu einem Webserver-Port her. Mit der Methode `XMLSocket.send()` übergeben Sie ein XML-Objekt an den in der Socketverbindung festgelegten Server.

Wenn Sie die `XMLSocket.connect()`-Methode aufrufen, öffnet die Anwendung eine TCP/IP-Verbindung zum Server und erhält diese Verbindung aufrecht, bis eine der folgenden Bedingungen eintritt:

- Die Methode `XMLSocket.close()` der XMLSocket-Klasse wird aufgerufen.
- Es sind keine Verweise auf das XMLSocket-Objekt mehr vorhanden.
- Flash Player wird beendet.
- Die Verbindung wird unterbrochen (z. B. wenn die Modemverbindung getrennt wird).

Herstellen einer Serververbindung mit der XMLSocket-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um eine Socketverbindung herzustellen, müssen Sie eine serverseitige Anwendung erstellen, die auf die Socketverbindungsanforderung wartet und eine Antwort an Flash Player oder die AIR-Anwendung sendet. Sie können diese serverseitige Anwendung in AIR oder in einer anderen Programmiersprache wie Java, Python oder Perl schreiben. Um die XMLSocket-Klasse verwenden zu können, muss auf dem Server ein Daemon ausgeführt werden, der das von der XMLSocket-Klasse verwendete einfache Protokoll verarbeiten kann:

- XML-Nachrichten werden über eine TCP/IP-Streaming-Socketverbindung im Vollduplexmodus gesendet.
- Jede XML-Nachricht ist ein vollständiges XML-Dokument, das mit einem Null-Byte (0) abgeschlossen wird.
- Es kann eine unbegrenzte Anzahl von XML-Nachrichten über eine einzelne XMLSocketverbindung gesendet und empfangen werden.

Erstellen und Herstellen einer Verbindung mit einem Java XML-Socket-Server

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der folgende Code zeigt, wie ein einfacher XMLSocket-Server in Java geschrieben wird, der ankommende Verbindungen akzeptiert und die empfangenen Nachrichten in einem Fenster mit Eingabeaufforderung anzeigt. In der Standardeinstellung wird ein neuer Server am Port 8080 Ihres lokalen Computers erstellt. Sie können jedoch beim Starten des Servers von der Befehlszeile auch eine andere Portnummer angeben.

Erstellen Sie ein neues Textdokument und geben Sie den folgenden Code ein:

```
import java.io.*;
import java.net.*;

class SimpleServer
{
    private static SimpleServer server;
    ServerSocket socket;
    Socket incoming;
    BufferedReader readerIn;
    PrintStream printOut;

    public static void main(String[] args)
    {
        int port = 8080;

        try
        {
            port = Integer.parseInt(args[0]);
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            // Catch exception and keep going.
        }

        server = new SimpleServer(port);
    }

    private SimpleServer(int port)
    {
        System.out.println(">> Starting SimpleServer");
        try
        {
            socket = new ServerSocket(port);
            incoming = socket.accept();
            readerIn = new BufferedReader(new InputStreamReader(incoming.getInputStream()));
            printOut = new PrintStream(incoming.getOutputStream());
            printOut.println("Enter EXIT to exit.\r");
            out("Enter EXIT to exit.\r");
            boolean done = false;
            while (!done)
            {
                String str = readerIn.readLine();
                if (str == null)
                {
                    done = true;
                }
```

```
                else
                {
                    out("Echo: " + str + "\r");
                    if(str.trim().equals("EXIT"))
                    {
                        done = true;
                    }
                }
                incoming.close();
            }
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }

    private void out(String str)
    {
        printOut.println(str);
        System.out.println(str);
    }
}
```

Speichern Sie das Dokument „SimpleServer.java“ auf der Festplatte und kompilieren Sie es mithilfe eines Java-Compilers, der eine Java-Klassendatei namens „SimpleServer.class“ erstellt.

Sie können den XMLSocket-Server starten, indem Sie eine Eingabeaufforderung öffnen und `java SimpleServer` eingeben. Die Datei „SimpleServer.class“ kann sich in einem beliebigen Verzeichnis auf dem lokalen Computer oder im Netzwerk befinden; sie muss nicht im Hauptverzeichnis des Webservers angelegt sein.

Wenn Sie den Server nicht starten können, da sich die Dateien nicht im Java-Klassenpfad befinden, versuchen Sie, den Server mit `java -classpath . SimpleServer` zu starten.

Um von der Anwendung aus eine Verbindung mit der XMLSocket-Klasse herzustellen, müssen Sie eine neue Instanz der XMLSocket-Klasse erstellen und die `XMLSocket.connect()`-Methode aufrufen, der Sie Hostnamen und Portnummer wie folgt übergeben:

```
var xmlsock:XMLSocket = new XMLSocket();
xmlsock.connect("127.0.0.1", 8080);
```

Jedes Mal, wenn Sie Daten vom Server empfangen, wird das `data`-Ereignis (`flash.events.DataEvent.DATA`) ausgelöst:

```
xmlsock.addEventListener(DataEvent.DATA, onData);
private function onData(event:DataEvent):void
{
    trace("[" + event.type + "] " + event.data);
}
```

Zum Senden von Daten an den XMLSocket-Server verwenden Sie die `XMLSocket.send()`-Methode und übergeben ein XML-Objekt oder einen String. Flash Player wandelt den angegebenen Parameter in ein String-Objekt um und sendet den Inhalt an den XMLSocket-Server, gefolgt von einem Null-Byte (0):

```
xmlsock.send(xmlFormattedData);
```

Die Methode `XMLSocket.send()` gibt keinen Wert zurück, der angibt, ob die Daten erfolgreich übermittelt wurden. Tritt bei dem Versuch, die Daten zu senden, ein Fehler auf, wird ein IOError-Fehler ausgelöst.

Jede Nachricht, die Sie an den XML-Socketserver senden, muss durch ein Zeilenvorschubzeichen (`\n`) beendet werden.

Weitere Informationen finden Sie unter XMLSocket.

Serversockets

Adobe AIR 2 und höher

Mithilfe der ServerSocket-Klasse geben Sie anderen Prozessen die Möglichkeit, über ein TCP-Socket (Transport Control Protocol) eine Verbindung mit Ihrer Anwendung herzustellen. Der Verbindungsprozess kann auf dem lokalen Computer oder auf einem anderen Computer im Netzwerk ausgeführt werden. Wenn ein ServerSocket-Objekt eine Verbindungsanforderung erhält, löst es ein `connect`-Ereignis aus. Das mit dem Ereignis ausgelöste ServerSocketConnectEvent-Objekt enthält ein Socket-Objekt. Dieses Socket-Objekt kann für die weitere Kommunikation mit dem anderen Prozess verwendet werden.

So warten Sie auf eingehende Socketverbindungen:

1. Erstellen Sie ein ServerSocket-Objekt und binden Sie es an einen lokalen Port.
2. Fügen Sie Ereignis-Listener für das `connect`-Ereignis hinzu.
3. Rufen Sie die `listen()`-Methode auf.
4. Reagieren Sie auf das `connect`-Ereignis, das ein Socket-Objekt für jede eingehende Verbindung bereitstellt.

Das ServerSocket-Objekt wartet auf neue Verbindungen, bis Sie die `close()`-Methode aufrufen.

Das folgende Codebeispiel veranschaulicht die Erstellung einer Socket-Serveranwendung. Im Beispiel wird an Port 8087 auf eingehende Verbindungen gewartet. Wenn eine Verbindung empfangen wird, sendet das Beispiel eine Nachricht (den String „Connected“) an das Clientsocket. Anschließend gibt der Server empfangene Nachrichten per Echo an den Client zurück.

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.IOErrorEvent;
    import flash.events.ProgressEvent;
    import flash.events.ServerSocketConnectEvent;
    import flash.net.ServerSocket;
    import flash.net.Socket;

    public class ServerSocketExample extends Sprite
    {
        private var serverSocket:ServerSocket;
        private var clientSockets:Array = new Array();

        public function ServerSocketExample()
        {
            try
            {
                // Create the server socket
                serverSocket = new ServerSocket();

                // Add the event listener
                serverSocket.addEventListener( Event.CONNECT, connectHandler );
                serverSocket.addEventListener( Event.CLOSE, onClose );
```

```
        // Bind to local port 8087
        serverSocket.bind( 8087, "127.0.0.1" );

        // Listen for connections
        serverSocket.listen();
        trace( "Listening on " + serverSocket.localPort );

    }
    catch(e:SecurityError)
    {
        trace(e);
    }
}

public function connectHandler(event:ServerSocketConnectEvent):void
{
    //The socket is provided by the event object
    var socket:Socket = event.socket as Socket;
    clientSockets.push( socket );

    socket.addEventListener( ProgressEvent.SOCKET_DATA, socketDataHandler);
    socket.addEventListener( Event.CLOSE, onClientClose );
    socket.addEventListener( IOErrorEvent.IO_ERROR, onIOError );

    //Send a connect message
    socket.writeUTFBytes("Connected.");
    socket.flush();

    trace( "Sending connect message" );
}

public function socketDataHandler(event:ProgressEvent):void
{
    var socket:Socket = event.target as Socket

    //Read the message from the socket
    var message:String = socket.readUTFBytes( socket.bytesAvailable );
    trace( "Received: " + message);

    // Echo the received message back to the sender
    message = "Echo -- " + message;
    socket.writeUTFBytes( message );
```

```
            socket.flush();
            trace( "Sending: " + message );
        }

        private function onClientClose( event:Event ):void
        {
            trace( "Connection to client closed." );
            //Should also remove from clientSockets array...
        }

        private function onIOError( errorEvent:IOErrorEvent ):void
        {
            trace( "IOError: " + errorEvent.text );
        }

        private function onClose( event:Event ):void
        {
            trace( "Server socket closed by OS." );
        }
}}
```

Weitere Informationen finden Sie unter:

- ServerSocket
- ServerSocketConnectEvent
- Socket

UDP-Sockets (AIR)

Adobe AIR 2 und höher

Das UDP-Protokoll (Universal Datagram Protocol) bietet die Möglichkeit, Nachrichten über eine Stateless-Netzwerkverbindung auszutauschen. Bei UDP besteht keine Garantie, dass Nachrichten zugestellt oder in der richtigen Reihenfolge zugestellt werden. Bei Verwendung von UDP wendet der Netzwerkcode des Betriebssystems meist weniger Zeit für Überwachung, Verfolgung und Bestätigung der Nachrichten auf. Deshalb gehen UDP-Nachrichten in der Regel mit einer kürzeren Verzögerung bei der Zielanwendung ein, als dies bei TCP-Nachrichten der Fall ist.

Die UDP-Socketkommunikation eignet sich gut zum Senden von Echtzeitinformationen, beispielsweise Positionsaktualisierungen in einem Spiel oder Soundpakete in einer Chat-Anwendung mit Audiofunktion. In solchen Anwendungen ist ein gewisses Maß an Datenverlust hinnehmbar, da eine niedrigere Übertragungslatenz wichtiger ist als die garantierte Nachrichtenzustellung. Für fast alle anderen Zwecke sind TCP-Sockets besser geeignet.

AIR-Anwendungen können UDP-Nachrichten über die DatagramSocket- und DatagramSocketDataEvent-Klassen senden und empfangen. So senden oder empfangen Sie eine UDP-Nachricht:

1. Erstellen Sie ein DatagramSocket-Objekt.
2. Fügen Sie einen Ereignis-Listener für das `data`-Ereignis hinzu.
3. Binden Sie das Socket mithilfe der `bind()`-Methode an eine lokale IP-Adresse und einen lokalen Port.
4. Senden Sie Nachrichten durch Aufruf der `send()`-Methode, wobei die IP-Adresse und der Port des Zielcomputers übergeben werden müssen.

5 Empfangen Sie Nachrichten durch Reagieren auf das `data`-Ereignis. Das für dieses Ereignis ausgelöste DatagramSocketDataEvent-Objekt enthält ein ByteArray-Objekt, das wiederum die Nachrichtendaten enthält.

Im folgenden Codebeispiel wird veranschaulicht, wie eine Anwendung UDP-Nachrichten senden und empfangen kann. Das Beispiel sendet eine einzelne Nachricht mit dem String „Hello." an den Zielcomputer. Außerdem wird der Inhalt der empfangenen Nachrichten verfolgt.

```
package
{
import flash.display.Sprite;
import flash.events.DatagramSocketDataEvent;
import flash.events.Event;
import flash.net.DatagramSocket;
import flash.utils.ByteArray;

public class DatagramSocketExample extends Sprite
{
    private var datagramSocket:DatagramSocket;

    //The IP and port for this computer
    private var localIP:String = "192.168.0.1";
    private var localPort:int = 55555;

    //The IP and port for the target computer
    private var targetIP:String = "192.168.0.2";
    private var targetPort:int = 55555;

    public function DatagramSocketExample()
    {
        //Create the socket
        datagramSocket = new DatagramSocket();
        datagramSocket.addEventListener( DatagramSocketDataEvent.DATA, dataReceived );

        //Bind the socket to the local network interface and port
        datagramSocket.bind( localPort, localIP );

        //Listen for incoming datagrams
        datagramSocket.receive();

        //Create a message in a ByteArray
        var data:ByteArray = new ByteArray();
        data.writeUTFBytes("Hello.");

        //Send the datagram message
        datagramSocket.send( data, 0, 0, targetIP, targetPort);
    }

    private function dataReceived( event:DatagramSocketDataEvent ):void
    {
        //Read the data from the datagram
        trace("Received from " + event.srcAddress + ":" + event.srcPort + "> " +
            event.data.readUTFBytes( event.data.bytesAvailable ) );
    }
}}
```

Berücksichtigen Sie bei Verwendung von UDP-Sockets die folgenden Punkte:

- Ein einzelnes Datenpaket kann nicht größer sein als die kleinste maximale Übertragungseinheit der Netzwerkschnittstelle oder eines Netzwerkknotens zwischen dem Sender und dem Empfänger. Alle Daten im ByteArray-Objekt, das an die send()-Methode übergeben wird, werden als einzelnes Paket gesendet. (Bei TCP werden umfangreiche Nachrichten in separate Pakete aufgeteilt.)
- Es erfolgt kein Handshaking zwischen Sender und Ziel. Wenn das Ziel nicht existiert oder keinen aktiven Listener am angegebenen Port hat, werden Nachrichten ohne Fehler verworfen.
- Bei Verwendung der `connect()`-Methode werden Nachrichten, die von anderen Quellen gesendet werden, ignoriert. Eine UDP-Verbindung bietet lediglich eine praktische Paketfilterung. Es ist nicht gewährleistet, dass sich an der Zieladresse und am Zielport ein gültiger Prozess befindet, der auf die Nachricht wartet.
- UDP-Datenverkehr kann zu einer Überbelastung des Netzwerks führen. Bei einer zu hohen Netzwerkbeanspruchung müssen Netzwerkadministratoren möglicherweise QoS-Steuerungen implementieren. (TCP verfügt über integrierte Datenverkehrssteuerungen, um die Belastung des Netzwerks zu verringern.)

Weitere Informationen finden Sie unter:

- DatagramSocket
- DatagramSocketDataEvent
- ByteArray

IPv6-Adressen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

IPv6 (Internet Protocol Version 6) wird ab Flash Player 9.0.115.0 unterstützt. IPv6 ist eine Internet Protocol-Version, die 128-Bit-Adressen unterstützt (eine Verbesserung gegenüber der älteren Version IPv4, die 32-Bit-Adressen unterstützt). Möglicherweise müssen Sie IPv6 für Ihre Netzwerkschnittstellen aktivieren. Weitere Informationen finden Sie in der Hilfe des Betriebssystems, unter dem die Daten gehostet werden.

Wenn IPv6 vom Hostingsystem unterstützt wird, können Sie in URLs numerische IPv6-Literaladressen in eckigen Klammern angeben, wie im folgenden Beispiel:

```
[2001:db8:ccc3:ffff:0:444d:555e:666f]
```

Flash Player gibt IPv6-Literaladressen entsprechende der folgenden Regeln zurück:

- Flash Player gibt die den String der IPv6-Adresse in der Langform zurück.
- Der IP-Wert weist keine Abkürzungen mit doppeltem Doppelpunkt auf.
- Hexadezimalwerte sind immer nur in Kleinschreibung.
- IPv6-Adressen sind in eckigen Klammern eingeschlossen.
- Jedes Adressquartett wird in Form von 0 bis 4 Hexadezimalwerten ausgegeben, wobei die führenden Nullen ausgelassen werden.
- Ein aus Nullen bestehendes Adressquartett wird als einzelne Null ausgegeben (nicht als doppelter Doppelpunkt); die Ausnahmen können Sie der folgenden Liste entnehmen.

Die von Flash Player zurückgegebenen IPv6-Werte weisen die folgenden Ausnahmen auf:

- Eine nicht angegebene IPv6-Adresse (ausschließlich Nullen) wird als [::] ausgegeben.

- Die Loopback- oder Localhost-IPv6-Adresse wird als [::1] ausgegeben.
- IPv4-zugeordnete Adressen (in IPv6 umgewandelt) werden als [::ffff:a.b.c.d] ausgegeben, wobei es sich bei „a.b.c.d“ um einen typischen Punktdezimalwert von IPv4 handelt.
- IPv4-kompatible Adressen werden als [::a.b.c.d] ausgegeben, wobei es sich bei „a.b.c.d“ um einen typischen Punktdezimalwert von IPv4 handelt.

Kapitel 44: HTTP-Kommunikation

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In Adobe® AIR® und Adobe® Flash® Player erstellte Anwendungen können mit HTTP-Servern kommunizieren, um Daten, Bilder und Video zu laden sowie Nachrichten auszutauschen.

Verwandte Hilfethemen

flash.net.URLLoader

flash.net.URLStream

flash.net.URLRequest

flash.net.URLRequestDefaults

flash.net.URLRequestHeader

flash.net.URLRequestMethod

flash.net.URLVariables

Laden von externen Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

ActionScript 3.0 umfasst Mechanismen zum Laden von Daten aus externen Quellen. Diese Quellen können statischen Inhalt, wie etwa Textdateien, oder dynamischen Inhalt, der von einem Webskript erzeugt wird, bereitstellen. Die Daten können auf verschiedene Weise formatiert sein, und ActionScript bietet Funktionen, um die Daten zu dekodieren und darauf zuzugreifen. Außerdem können Sie als Teil des Datenabrufverfahrens Daten an den externen Server senden.

Verwenden der URLRequest-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Viele APIs, die externe Daten laden, verwenden die URLRequest-Klasse, um die Eigenschaften der erforderlichen Netzwerkanforderung zu definieren.

URLRequest-Eigenschaften

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die folgenden Eigenschaften eines URLRequest-Objekts können in jeder Sicherheits-Sandbox festgelegt werden:

Eigenschaft	Beschreibung
contentType	Der MIME-Inhaltstyp von Daten, die mit der URL-Anforderung gesendet wurden. Wenn kein Wert für „contentType“ eingestellt ist, werden Werte als `application/x-www-form-urlencoded` gesendet.
data	Ein Objekt, das mit der URL-Anforderung zu übertragende Daten enthält.
digest	Ein String, der die vorzeichenbehaftete Adobe-Plattformkomponente, die im Adobe® Flash® Player-Cache gespeichert (bzw. daraus abgerufen) werden soll, eindeutig identifiziert.
method	Die HTTP-Anforderungsmethode, wie beispielsweise GET oder POST: (Der in der Sicherheitsdomäne der AIR-Anwendung ausgeführte Inhalt kann andere Strings als `"GET"` oder `"POST"` für die `method`-Eigenschaft einstellen. Jedes HTTP-Verb ist zulässig, und `"GET"` ist die Standardmethode. Siehe „AIR-Sicherheit“ auf Seite 1153.)
requestHeaders	Das Array der an die HTTP-Anforderung anzuhängenden HTTP-Anforderungsheader. Beachten Sie, dass das Einstellen der Header in Flash Player und in AIR-Inhalt, der außerhalb der Anwendungssicherheits-Sandbox ausgeführt wird, Beschränkungen unterliegt.
url	Gibt die anzufordernde URL an.

In AIR können Sie zusätzliche Eigenschaften der URLRequest-Klasse festlegen, die nur für AIR-Inhalt zur Verfügung stehen, der in der Anwendungssicherheits-Sandbox ausgeführt wird. URLs von Inhalt in der Anwendungs-Sandbox können mithilfe neuer URL-Schemas (zusätzlich zu Standardschemas wie `file` und `http`) definiert werden.

Eigenschaft	Beschreibung
followRedirects	Legt fest, ob Weiterleitungen befolgt werden (`true`, der Standardwert) oder nicht (`false`). Dies wird nur in der AIR-Anwendungs-Sandbox unterstützt.
manageCookies	Gibt an, ob der HTTP-Protokollstapel bei dieser Anforderung Cookies verwalten soll (`true`, der Standardwert) oder nicht (`false`). Das Festlegen dieser Eigenschaft wird nur in der AIR-Anwendungs-Sandbox unterstützt.
authenticate	Legt fest, ob Authentifizierungsanforderungen für diese Anforderung verarbeitet werden (`true`). Das Festlegen dieser Eigenschaft wird nur in der AIR-Anwendungs-Sandbox unterstützt. Anforderungen werden standardmäßig authentifiziert. Falls der Server die Anzeige von Benutzerangaben erfordert, wird ein Dialogfeld zur Authentifizierung geöffnet. Sie können auch den Benutzernamen und das Kennwort mit der URLRequestDefaults-Klasse festlegen; siehe „Festlegen der URLRequest-Standardeinstellungen (nur AIR)“ auf Seite 866.
cacheResponse	Gibt an, ob Antwortdaten für diese Anforderung zwischengespeichert werden sollen. Das Festlegen dieser Eigenschaft wird nur in der AIR-Anwendungs-Sandbox unterstützt. Die Standardeinstellung ist, die Antwort zwischenzuspeichern (`true`).
useCache	Legt fest, ob der lokale Cache überprüft wird, bevor diese URLRequest Daten abruft. Das Festlegen dieser Eigenschaft wird nur in der AIR-Anwendungs-Sandbox unterstützt. Die Standardeinstellung lautet `true`, d. h. falls verfügbar wird die lokal zwischengespeicherte Version wird verwendet.
userAgent	Legt den Benutzer-Agent-String fest, der in der HTTP-Anforderung verwendet wird.

Hinweis: *Die HTMLLoader-Klasse bietet ähnliche Eigenschaften für Einstellungen, die sich auf Inhalt beziehen, der mit einem HTMLLoader-Objekt geladen wurde. Weitere Informationen finden Sie unter „HTMLLoader-Klasse“ auf Seite 1047.*

Festlegen der URLRequest-Standardeinstellungen (nur AIR)

Adobe AIR 1.0 und höher

Mithilfe der URLRequestDefaults-Klasse können Sie anwendungsspezifische Standardeinstellungen für URLRequest-Objekte definieren. Im folgenden Beispiel werden die Standardwerte für die `manageCookies`- und `useCache`-Eigenschaften festgelegt. Für alle neuen URLRequest-Objekte werden die angegebenen Werte dieser Eigenschaften anstelle der regulären Standardeinstellungen verwendet:

```
URLRequestDefaults.manageCookies = false;
URLRequestDefaults.useCache = false;
```

Hinweis: *Die URLRequestDefaults-Klasse ist nur für Inhalt, der in Adobe AIR ausgeführt wird, definiert. In Flash Player gibt es dafür keine Unterstützung.*

Die URLRequestDefaults-Klasse enthält die `setLoginCredentialsForHost()`-Methode, mit der Sie einen standardmäßigen Benutzernamen und ein Kennwort für einen bestimmten Host festlegen können. Dieser Host, der im hostname-Parameter der Methode definiert ist, kann eine Domäne sein, zum Beispiel `"www.example.com"` oder eine Domäne und eine Portnummer, zum Beispiel `"www.example.com:80"`. Beachten Sie, dass `"example.com"`, `"www.example.com"` und `"sales.example.com"` jeweils als eindeutige Hosts angesehen werden.

Diese Benutzerangaben werden nur verwendet, wenn sie der Server erforderlich macht. Wenn der Benutzer bereits authentifiziert wurde (zum Beispiel über das Dialogfeld für die Authentifizierung), wird durch Aufruf der `setLoginCredentialsForHost()`-Methode der authentifizierte Benutzer nicht geändert.

Mit dem folgenden Code werden der Standardbenutzername und das Kennwort für Anforderungen an www.example.com festgelegt:

```
URLRequestDefaults.setLoginCredentialsForHost("www.example.com", "Ada", "love1816$X");
```

Die URLRequestDefaults-Einstellungen gelten nur für die aktuelle Anwendungsdomäne, mit einer Ausnahme. Die an die `setLoginCredentialsForHost()`-Methode übergebenen Benutzerinformationen werden für Anforderungen in jeder Anwendungsdomäne innerhalb der AIR-Anwendung benutzt.

Weitere Informationen finden Sie in der Beschreibung der URLRequestDefaults-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

URI-Schemas

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die URI-Standardschemas, wie beispielsweise die folgenden, können in Anforderungen von jeder Sicherheits-Sandbox verwendet werden:

http: und https:
Verwenden Sie diese Schemas für standardmäßige Internet-URLs (genau wie bei der Verwendung in einem Webbrowser).

file:
Verwenden Sie `file:` zur Angabe der URL einer Datei, die sich im lokalen Dateisystem befindet. Zum Beispiel:

```
file:///c:/AIR Test/test.txt
```

In AIR können Sie auch eines der folgenden Schemas verwenden, wenn Sie eine URL für Inhalt definieren, der in der Sicherheits-Sandbox der Anwendung ausgeführt wird:

app:
Geben Sie mit `app:` einen Pfad relativ zum Stamm der installierten Anwendung an. Beispielsweise verweist der folgende Pfad auf ein „resources“ genanntes Unterverzeichnis im Verzeichnis der installierten Anwendung:

```
app:/resources
```

Wenn eine AIR-Anwendung über ADL (AIR Debug Launcher) gestartet wird, ist das Anwendungsverzeichnis das Verzeichnis, das die Deskriptordatei der Anwendung enthält.

Die URL (und die `url`-Eigenschaft) eines mit `File.applicationDirectory` erstellten File-Objekts verwenden das URI-Schema `app`, wie im Folgenden gezeigt:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("assets");
trace(dir.url); // app:/assets
```

app-storage:

Geben Sie mit `app-storage:` einen Pfad relativ zum Datenspeicherverzeichnis der installierten Anwendung an. Für jede installierte Anwendung (und jeden Benutzer) erstellt AIR ein eindeutiges Anwendungsspeicherverzeichnis, ein nützlicher Ort zum Speichern anwendungsspezifischer Daten. Der folgende Pfad etwa verweist von einem File-Objekt auf eine Datei mit dem Namen „prefs.xml“ in einem settings-Verzeichnis des Anwendungsspeicherverzeichnisses:

```
app-storage:/settings/prefs.xml
```

Die URL (und die `url`-Eigenschaft) eines mit `File.applicationStorageDirectory` erstellten File-Objekts verwenden das URI-Schema `app-storage`, wie im Folgenden gezeigt:

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("prefs.xml");
trace(dir.prefsFile); // app-storage:/prefs.xml
```

mailto:

Sie können das mailto-Schema in URLRequest-Objekten verwenden, die an die `navigateToURL()`-Funktion übergeben werden. Siehe „Öffnen einer URL in einer anderen Anwendung“ auf Seite 882.

Mit einem URLRequest-Objekt, das eines dieser URI-Schemas verwendet, können Sie die URL-Anforderung für mehrere verschiedene Objekte, wie etwa ein FileStream- oder ein Sound-Objekt, definieren. Sie können diese Schemas auch mit HTML-Inhalt, der in AIR ausgeführt wird, verwenden; z. B. im `src`-Attribut eines `img`-Tags.

Diese AIR-spezifischen URI-Schemas (`app:` und `app-storage:`) können Sie jedoch nur in Inhalt in der Sicherheits-Sandbox der Anwendung verwenden. Weitere Informationen finden Sie unter „AIR-Sicherheit“ auf Seite 1153.

Einstellen von URL-Variablen

Variablen können zwar direkt in einem URL-String hinzugefügt werden, doch möglicherweise ist es einfacher, die für eine Anforderung erforderlichen Variablen über die URLVariables-Klasse zu definieren.

Es gibt drei Möglichkeiten, um einem URLVariables-Objekt Parameter hinzuzufügen:

- Im URLVariables-Konstruktor
- Mit der `URLVariables.decode()`-Methode
- Als dynamische Eigenschaften des URLVariables-Objekts selbst

Das folgende Beispiel veranschaulicht alle drei Möglichkeiten und zeigt auch, wie die Variablen einem URLRequest-Objekt zugewiesen werden:

```
var urlVar:URLVariables = new URLVariables( "one=1&two=2" );
urlVar.decode("amp=" + encodeURIComponent( "&" ) );
urlVar.three = 3;
urlVar.amp2 = "&&";
trace(urlVar.toString()); //amp=%26&amp2=%26%26&one=1&two=2&three=3

var urlRequest:URLRequest = new URLRequest( "http://www.example.com/test.cfm" );
urlRequest.data = urlVar;
```

Bei der Definition von Variablen innerhalb des URLVariables-Konstruktors oder innerhalb der `URLVariables.decode()`-Methode müssen die Zeichen, die in einem URI-String eine besondere Bedeutung haben, in URL-Kodierung formatiert werden. Wenn ein Parametername oder ein Parameterwert beispielsweise das &-Zeichen enthält, müssen Sie das `&` als `%26` kodieren, da dieses Zeichen als Trennzeichen für Parameter dient. Zu diesem Zweck kann die `encodeURIComponent()`-Funktion auf oberster Ebene verwendet werden.

Verwenden der URLLoader-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der URLLoader-Klasse können Sie eine Anforderung an einen Server senden und auf die zurückgegebenen Informationen zugreifen. Außerdem können Sie mithilfe der URLLoader-Klasse auf Dateien im lokalen Dateisystem zugreifen, wenn der lokale Dateizugriff zulässig ist (beispielsweise in der Sandbox „local-with-filesystem“ von Flash Player oder in der Sandbox der AIR-Anwendung). Die URLLoader-Klasse lädt die Daten von einer URL als Text, Binärdaten oder URL-kodierte Variablen herunter. Die URLLoader-Klasse löst verschiedene Ereignisse aus, wie zum Beispiel `complete`, `httpStatus`, `ioError`, `open`, `progress` und `securityError`.

Das Ereignisverarbeitungsmodell in ActionScript 3.0 unterscheidet sich ganz erheblich von dem ActionScript 2.0-Modell, das die Ereignisprozeduren `LoadVars.onData`, `LoadVars.onHTTPStatus` und `LoadVars.onLoad` verwendet. Weitere Informationen zur Ereignisverarbeitung in ActionScript 3.0 finden Sie unter „Verarbeiten von Ereignissen“ auf Seite 133.

Die heruntergeladenen Daten stehen erst zur Verfügung, wenn der Download abgeschlossen ist. Sie können den Fortschritt des Downloads überwachen (geladene Byte und Byte gesamt), indem Sie auf die Auslösung des `progress`-Ereignisses warten. Doch wenn eine Datei sehr schnell geladen wird, wird möglicherweise kein `progress`-Ereignis ausgelöst. Sobald eine Datei erfolgreich heruntergeladen wurde, wird das `complete`-Ereignis ausgelöst. Durch Einstellen der URLLoader-Eigenschaft `dataFormat` können Sie festlegen, dass Daten als Text, in Form von unformatierten Binärdaten oder als URLVariables-Objekt empfangen werden.

Die `URLLoader.load()`-Methode (und wahlweise der Konstruktor der URLLoader-Klasse) arbeiten mit einem Parameter, `request`, bei dem es sich um ein URLRequest-Objekt handelt. Ein URLRequest-Objekt enthält alle Informationen einer HTTP-Anforderung, beispielsweise die Ziel-URL, die Anforderungsmethode (`GET` oder `POST`), zusätzliche Header-Informationen sowie den MIME-Typ.

Im folgenden Beispiel wird ein XML-Paket zu einem serverseitigen Skript hochgeladen:

```
var secondsUTC:Number = new Date().time;
var dataXML:XML =
    <clock>
        <time>{secondsUTC}</time>
    </clock>;
var request:URLRequest = new URLRequest("http://www.yourdomain.com/time.cfm");
request.contentType = "text/xml";
request.data = dataXML.toXMLString();
request.method = URLRequestMethod.POST;
var loader:URLLoader = new URLLoader();
loader.load(request);
```

Mit diesem Codefragment wird ein XML-Dokument namens `dataXML` erstellt, welches das XML-Paket enthält, das an den Server gesendet wird. In diesem Beispiel wird die URLRequest-Eigenschaft `contentType` auf `"text/xml"` eingestellt und das XML-Dokument wird der URLRequest-Eigenschaft `data` zugewiesen. Abschließend erstellt das Beispiel ein URLLoader-Objekt und sendet die Anforderung über die `load()`-Methode an das Remote-Skript.

Verwenden der URLStream-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die URLStream-Klasse ermöglicht den Zugriff auf die heruntergeladenen Daten direkt beim Datenempfang. Mit der URLStream-Klasse können Sie außerdem einen Stream beenden, bevor der Download abgeschlossen ist. Die heruntergeladenen Daten stehen als unformatierte Binärdaten zur Verfügung.

Verwenden Sie beim Lesen von Daten aus einem URLStream-Objekt die `bytesAvailable`-Eigenschaft, um festzustellen, ob vor dem Lesen ausreichend Daten zur Verfügung stehen. Eine EOFError-Ausnahme wird ausgelöst, wenn Sie versuchen, mehr Daten zu lesen als derzeit verfügbar sind.

Das httpResponseStatus-Ereignis (AIR)

In Adobe AIR löst die URLStream-Klasse ein `httpResponseStatus`-Ereignis zusätzlich zum `httpStatus`-Ereignis aus. Das `httpResponseStatus`-Ereignis wird ausgelöst, bevor Antwortdaten empfangen werden. Das `httpResponseStatus`-Ereignis (das von der HTTPStatusEvent-Klasse dargestellt wird) enthält eine `responseURL`-Eigenschaft (die URL, die die Antwort zurückgegeben hat) und eine `responseHeaders`-Eigenschaft (ein Array von URLRequestHeader-Objekten, die die Antwort-Header der zurückgegebenen Antwort darstellen).

Laden von Daten aus externen Dokumenten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie dynamische Anwendungen erstellen, kann es hilfreich sein, Daten aus externen Dateien oder aus serverseitigen Skripts zu laden. Auf diese Weise können Sie dynamische Anwendungen erstellen, ohne dass Sie Ihre Anwendung bearbeiten oder neu kompilieren müssen. Angenommen, Sie erstellen eine Anwendung „Tipp des Tages“. In diesem Fall können Sie ein serverseitiges Skript erstellen, das einen zufälligen Tipp aus einer Datenbank abruft und einmal am Tag in einer Textdatei speichert. Dann kann Ihre Anwendung die Inhalte einer statischen Textdatei laden, anstatt jedes Mal die Datenbank abzufragen.

Der folgende Codeausschnitt erstellt ein URLRequest- und ein URLLoader-Objekt, welche die Inhalte einer externen Textdatei namens „params.txt“ laden:

```
var request:URLRequest = new URLRequest("params.txt");
var loader:URLLoader = new URLLoader();
loader.load(request);
```

In der Standardeinstellung laden Flash Player und Adobe AIR die Inhalte mithilfe der Methode `HTTP GET`, wenn Sie keine Anforderungsmethode definieren. Um die Anforderung mithilfe der `POST`-Methode zu senden, geben Sie für die `request.method`-Eigenschaft den Wert `POST` mittels der statischen Konstante `URLRequestMethod.POST` an. Siehe hierzu auch das folgende Beispiel:

```
var request:URLRequest = new URLRequest("sendfeedback.cfm");
request.method = URLRequestMethod.POST;
```

Das externe Dokument „params.txt“, das zur Laufzeit geladen wird, enthält die folgenden Daten:

```
monthNames=January,February,March,April,May,June,July,August,September,October,November,Dece
mber&dayNames=Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

Die Datei enthält zwei Parameter: `monthNames` und `dayNames`. Jeder Parameter enthält eine durch Kommas getrennte Liste, die als Strings geparst wird. Sie können diese Liste mit der `String.split()`-Methode in einen Array aufteilen.

Vermeiden Sie das Verwenden von reservierten Wörtern oder Sprachkonstrukten in externen Datendateien, da andernfalls das Lesen und Debuggen Ihres Codes schwieriger wird.

Nachdem die Daten geladen wurden, wird das `complete`-Ereignis ausgelöst, und der Inhalt des externen Dokuments steht der URLLoader-Eigenschaft `data` zur Verfügung. Dies wird im folgenden Code gezeigt:

```
function completeHandler(event)
{
    var loader2 = event.target;
    air.trace(loader2.data);
}
```

Wenn das Remote-Dokument Name-Wert-Paare enthält, können Sie die Daten mit der URLVariables-Klasse parsen, indem Sie den Inhalt der geladenen Datei wie folgt übergeben:

```
private function completeHandler(event:Event):void
{
    var loader2:URLLoader = URLLoader(event.target);
    var variables:URLVariables = new URLVariables(loader2.data);
    trace(variables.dayNames);
}
```

Jedes Name-Wert-Paare aus der externen Datei wird als eine Eigenschaft im URLVariables-Objekt erstellt. Im vorausgehenden Beispiel wird jede Eigenschaft im Variablenobjekt wie ein String behandelt. Handelt es sich beim Wert des Name-Wert-Paars um eine Liste mit Elementen, können Sie den String durch Aufrufen der `String.split()`-Methode wie folgt in ein Array umwandeln:

```
var dayNameArray:Array = variables.dayNames.split(",");
```

Wenn Sie numerische Daten aus externen Textdateien laden, wandeln Sie die Werte mithilfe einer Funktion auf oberster Ebene, wie z. B. `int()`, `uint()` oder `Number()`, in numerische Werte um.

Alternativ zum Laden der Inhalte der externen Datei als String und dem Erstellen eines neuen URLVariables-Objekts können Sie die `URLLoader.dataFormat`-Eigenschaft auf eine der statischen Eigenschaften der URLLoaderDataFormat-Klasse einstellen. Die drei möglichen Werte der `URLLoader.dataFormat`-Eigenschaft lauten:

- `URLLoaderDataFormat.BINARY` – Die Eigenschaft `URLLoader.data` enthält Binärdaten, die in einem ByteArray-Objekt gespeichert sind.
- `URLLoaderDataFormat.TEXT` – Die Eigenschaft `URLLoader.data` enthält Text in einem String-Objekt.
- `URLLoaderDataFormat.VARIABLES` – Die Eigenschaft `URLLoader.data` enthält URL-kodierte Variablen, die in einem URLVariables-Objekt gespeichert sind.

Der folgende Code zeigt, wie das Einstellen der Eigenschaft `URLLoader.dataFormat` auf `URLLoaderDataFormat.VARIABLES` es Ihnen ermöglicht, geladene Daten automatisch in ein URLVariables-Objekt zu analysieren:

```
package
{
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class URLLoaderDataFormatExample extends Sprite
    {
        public function URLLoaderDataFormatExample()
        {
            var request:URLRequest = new URLRequest("http://www.[yourdomain].com/params.txt");
            var variables:URLLoader = new URLLoader();
            variables.dataFormat = URLLoaderDataFormat.VARIABLES;
            variables.addEventListener(Event.COMPLETE, completeHandler);
            try
            {
                variables.load(request);
            }
            catch (error:Error)
            {
                trace("Unable to load URL: " + error);
            }
        }
        private function completeHandler(event:Event):void
        {
        var loader:URLLoader = URLLoader(event.target);
        trace(loader.data.dayNames);
        }
    }
}
```

Hinweis: *Der Standardwert für* `URLLoader.dataFormat` *ist* `URLLoaderDataFormat.TEXT`.

Das folgende Beispiel zeigt, dass XML aus einer externen Datei auf die gleiche Weise geladen wird wie URLVariables. Sie können eine URLRequest-Instanz und eine URLLoader-Instanz erstellen, um mit ihnen ein remotes XML-Dokument herunterzuladen. Nachdem die Datei vollständig heruntergeladen wurde, wird das `Event.COMPLETE`-Ereignis ausgelöst, und die Inhalte der externen Datei werden in eine XML-Instanz umgewandelt, die Sie mit XML-Methoden und -Eigenschaften analysieren können.

```
package
{
    import flash.display.Sprite;
    import flash.errors.*;
    import flash.events.*;
    import flash.net.URLLoader;
    import flash.net.URLRequest;

    public class ExternalDocs extends Sprite
    {
        public function ExternalDocs()
        {
            var request:URLRequest = new URLRequest("http://www.[yourdomain].com/data.xml");
            var loader:URLLoader = new URLLoader();
            loader.addEventListener(Event.COMPLETE, completeHandler);
            try
            {
                loader.load(request);
            }
            catch (error:ArgumentError)
            {
                trace("An ArgumentError has occurred.");
            }
            catch (error:SecurityError)
            {
                trace("A SecurityError has occurred.");
            }
        }
        private function completeHandler(event:Event):void
        {
            var dataXML:XML = XML(event.target.data);
            trace(dataXML.toXMLString());
        }
    }
}
```

Kommunizieren mit externen Skripts

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können die URLVariables-Klasse neben dem Laden von externen Datendateien auch zum Senden von Variablen an ein serverseitiges Skript und zum Bearbeiten der Antwort des Servers verwenden. Dies ist z. B. nützlich, wenn Sie ein Spiel programmieren und das Ergebnis des Benutzers an einen Server senden möchten, um zu berechnen, ob es zur Bestenliste hinzugefügt werden sollte, oder um die Anmeldeinformationen des Benutzers zur Validierung an einen Server zu senden. Ein serverseitiges Skript kann den Benutzernamen und das Kennwort verarbeiten, kann sie anhand einer Datenbank validieren und ggf. die Gültigkeit der vom Benutzer eingegebenen Benutzerangaben bestätigen.

Der folgende Codeausschnitt erstellt ein URLVariables-Objekt namens `variables`, das ein neues Objekt mit der Bezeichnung `name` erstellt. Als Nächstes wird ein URLRequest-Objekt erstellt, mit dem die URL des serverseitigen Skripts, an das die Variablen gesendet werden sollen, angegeben wird. Dann legen Sie mit der `method`-Eigenschaft des URLRequest-Objekts fest, dass die Variablen als eine `POST`-Anforderung von HTTP gesendet werden. Um das URLVariables-Objekt zur URL-Anforderung hinzuzufügen, stellen Sie die `data`-Eigenschaft des URLRequest-Objekts auf das zuvor erstellte URLVariables-Objekt ein. Daraufhin wird die URLLoader-Instanz erstellt, und die `URLLoader.load()`-Methode wird aufgerufen, die die Anforderung einleitet.

```
var variables:URLVariables = new URLVariables("name=Franklin");
var request:URLRequest = new URLRequest();
request.url = "http://www.[yourdomain].com/greeting.cfm";
request.method = URLRequestMethod.POST;
request.data = variables;
var loader:URLLoader = new URLLoader();
loader.dataFormat = URLLoaderDataFormat.VARIABLES;
loader.addEventListener(Event.COMPLETE, completeHandler);
try
{
    loader.load(request);
}
catch (error:Error)
{
    trace("Unable to load URL");
}

function completeHandler(event:Event):void
{
    trace(event.target.data.welcomeMessage);
}
```

Der folgende Code enthält den Inhalt des Adobe ColdFusion®-Dokuments „greeting.cfm“, das im vorhergehenden Beispiel verwendet wurde:

```
<cfif NOT IsDefined("Form.name") OR Len(Trim(Form.Name)) EQ 0>
    <cfset Form.Name = "Stranger" />
</cfif>
<cfoutput>welcomeMessage=#UrlEncodedFormat("Welcome, " & Form.name)#
</cfoutput>
```

Webdienstanforderungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Es gibt verschiedene HTTP-basierte Webdienste. Zu den Haupttypen zählen:

- REST
- XML-RPC
- SOAP

Zur Verwendung eines Webdienstes in ActionScript 3 erstellen Sie ein URLRequest-Objekt, konstruieren den Aufruf des Webdienstes entweder mithilfe von URL-Variablen oder eines XML-Dokuments und senden den Aufruf dann über ein URLLoader-Objekt an den Dienst. Das Flex Framework enthält mehrere Klassen zur einfacheren Verwendung von Webdiensten, die besonders beim Zugriff auf komplexe SOAP-Dienste hilfreich sind. Ab Flash Professional CS3 können Sie die Flex-Klassen in Anwendungen verwenden, die mit Flash Professional oder mit Flash Builder entwickelt wurden.

In HTML-basierten AIR-Anwendungen können Sie entweder die URLRequest- und URLLoader-Klassen oder die XMLHttpRequest-Klasse aus JavaScript verwenden. Bei Bedarf können Sie auch eine SWF-Bibliothek erstellen, die die Webdienstkomponenten im Flex Framework für Ihren JavaScript-Code zur Verfügung stellt.

Wenn Ihre Anwendung in einem Browser ausgeführt wird, können Sie nur Webdienste verwenden, die sich in derselben Internetdomäne befinden wie die aufrufende SWF-Datei, es sei denn, auf dem Server, der den Webdienst hostet, befindet sich auch eine domänenübergreifende Richtliniendatei, die den Zugriff von anderen Domänen aus zulässt. Wenn keine domänenübergreifende Richtliniendatei verfügbar ist, werden die Anforderungen als Alternativlösung häufig per Proxy durch den eigenen Server geleitet. Adobe Blaze DS und Adobe LiveCycle unterstützen die Proxy-Weiterleitung von Webdiensten.

In AIR-Anwendungen ist keine domänenübergreifende Richtliniendatei erforderlich, wenn der Webdienst von der Sicherheits-Sandbox der Anwendung aus aufgerufen wird. Der Inhalt von AIR-Anwendungen wird grundsätzlich nicht von Remote-Domänen bereitgestellt und ist deshalb immun gegenüber den Angriffen, die domänenübergreifende Richtlinien verhindern. In HTML-basierten AIR-Anwendungen kann Inhalt in der Sicherheits-Sandbox der Anwendung domänenübergreifende XMLHttpRequests-Anforderungen durchführen. Sie können dem Inhalt in anderen Sicherheits-Sandboxen die Durchführung von domänenübergreifenden XMLHttpRequests-Anforderungen erlauben, vorausgesetzt, der Inhalt ist in einem iframe geladen.

Verwandte Hilfethemen

„Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125

Adobe BlazeDS

Adobe LiveCycle ES2

REST-Architektur

XML-RPC

SOAP-Protokoll

REST-Webdienstanforderungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In REST-Webdienstanforderungen werden HTTP-Methodenverben verwendet, um die grundlegende Aktion festzulegen, und URL-Variablen, um die Aktionsdetails zu definieren. In einer Anforderung zum Abrufen von Daten für ein Element könnten beispielsweise das GET-Verb und URL-Variablen verwendet werden, um einen Methodennamen und die Element-ID anzugeben. Der resultierende URL-String könnte ungefähr folgendermaßen aussehen:

```
http://service.example.com/?method=getItem&id=d3452
```

Zum Zugriff auf REST-Webdienste mit ActionScript können Sie die URLRequest-, URLVariables- und URLLoader-Klassen verwenden. In JavaScript-Code innerhalb einer AIR-Anwendung kann auch eine XMLHttpRequest-Anforderung verwendet werden.

Zum Programmieren des Aufrufs eines REST-Webdienstes in ActionScript sind normalerweise die folgenden Schritte erforderlich:

1. Erstellen Sie ein URLRequest-Objekt.
2. Stellen Sie die Dienst-URL und das HTTP-Methodenverb für das Anforderungsobjekt ein.
3. Erstellen Sie ein URLVariables-Objekt.
4. Stellen Sie die Parameter für den Dienstaufruf als dynamische Eigenschaften des Variablenobjekts ein.
5. Weisen Sie das Variablenobjekt der Dateneigenschaft des Anforderungsobjekts zu.
6. Senden Sie den Aufruf mit einem URLLoader-Objekt an den Dienst.

7 Verarbeiten Sie das von URLLoader ausgelöste `complete`-Ereignis, das darauf hinweist, dass der Dienstaufruf abgeschlossen ist. Es empfiehlt sich auch, auf die verschiedenen Fehlerereignisse zu warten, die von einem URLLoader-Objekt ausgelöst werden können.

Angenommen, ein Webdienst stellt eine Testmethode bereit, die die Aufrufparameter per Echo an das aufrufende Objekt zurückgibt. Der folgende ActionScript-Code könnte zum Aufruf des Dienstes verwendet werden:

```
import flash.events.Event;
import flash.events.ErrorEvent;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLLoader;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLVariables;

private var requestor:URLLoader = new URLLoader();
public function restServiceCall():void
{
    //Create the HTTP request object
    var request:URLRequest = new URLRequest( "http://service.example.com/" );
    request.method = URLRequestMethod.GET;

    //Add the URL variables
    var variables:URLVariables = new URLVariables();
    variables.method = "test.echo";
    variables.api_key = "123456ABC";
    variables.message = "Able was I, ere I saw Elba.";
    request.data = variables;

    //Initiate the transaction
    requestor = new URLLoader();
    requestor.addEventListener( Event.COMPLETE, httpRequestComplete );
    requestor.addEventListener( IOErrorEvent.IOERROR, httpRequestError );
    requestor.addEventListener( SecurityErrorEvent.SECURITY_ERROR, httpRequestError );
    requestor.load( request );
}
private function httpRequestComplete( event:Event ):void
{
    trace( event.target.data );
}

private function httpRequestError( error:ErrorEvent ):void{
    trace( "An error occured: " + error.message );
}
```

In JavaScript innerhalb einer AIR-Anwendung kann dieselbe Anforderung mit dem XMLHttpRequest-Objekt durchgeführt werden:

```
<html>
<head><title>RESTful web service request</title>
<script type="text/javascript">

function makeRequest()
{
    var requestDisplay = document.getElementById( "request" );
    var resultDisplay  = document.getElementById( "result" );

    //Create a conveninece object to hold the call properties
    var request = {};
    request.URL = "http://service.example.com/";
    request.method = "test.echo";
    request.HTTPmethod = "GET";
    request.parameters = {};
    request.parameters.api_key = "ABCDEF123";
    request.parameters.message = "Able was I ere I saw Elba.";
    var requestURL = makeURL( request );
    xmlhttp = new XMLHttpRequest();
    xmlhttp.open( request.HTTPmethod, requestURL, true);
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            resultDisplay.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);

    requestDisplay.innerHTML = requestURL;
}
//Convert the request object into a properly formatted URL
function makeURL( request )
{
    var url = request.URL + "?method=" + escape( request.method );
    for( var property in request.parameters )
    {
        url += "&" + property + "=" + escape( request.parameters[property] );
    }

    return url;
}
</script>
</head>
<body onload="makeRequest()">
<h1>Request:</h1>
<div id="request"></div>
<h1>Result:</h1>
<div id="result"></div>
</body>
</html>
```

XML-RPC-Webdienstanforderungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei einem XML-RPC-Webdienst stammen die Aufrufparameter aus einem XML-Dokument, nicht aus URL-Variablen. Um eine Transaktion mit einem XML-RPC-Webdienst durchzuführen, erstellen Sie eine korrekt formatierte XML-Nachricht und senden Sie sie über die HTTP-Methode `POST` an den Webdienst. Außerdem sollten Sie den `Content-Type`-Header für die Anforderung festlegen, damit der Server die Anforderungsdaten als XML verarbeitet.

Das folgende Beispiel zeigt den Webdienstaufruf aus dem REST-Beispiel, dieses Mal jedoch als XML-RPC-Dienst:

```
import flash.events.Event;
import flash.events.ErrorEvent;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLLoader;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLVariables;
public function xmlRPCRequest():void
{
    //Create the XML-RPC document
    var xmlRPC:XML = <methodCall>
                                    <methodName></methodName>
                                    <params>
                                        <param>
                                            <value>
                                                <struct/>
                                            </value>
                                        </param>
                                    </params>
                                </methodCall>;

    xmlRPC.methodName = "test.echo";

    //Add the method parameters
    var parameters:Object = new Object();
    parameters.api_key = "123456ABC";
    parameters.message = "Able was I, ere I saw Elba.";

    for( var propertyName:String in parameters )
    {
        xmlRPC..struct.member[xmlRPC..struct.member.length + 1] =
            <member>
                <name>{propertyName}</name>
                <value>
                    <string>{parameters[propertyName]}</string>
                </value>
            </member>;
    }

    //Create the HTTP request object
    var request:URLRequest = new URLRequest( "http://service.example.com/xml-rpc/" );
    request.method = URLRequestMethod.POST;
    request.cacheResponse = false;
    request.requestHeaders.push(new URLRequestHeader("Content-Type", "application/xml"));
```

```
    request.data = xmlRPC;

    //Initiate the request
    requestor = new URLLoader();
    requestor.dataFormat = URLLoaderDataFormat.TEXT;
    requestor.addEventListener( Event.COMPLETE, xmlRPCRequestComplete );
    requestor.addEventListener( IOErrorEvent.IO_ERROR, xmlRPCRequestError );
    requestor.addEventListener( SecurityErrorEvent.SECURITY_ERROR, xmlRPCRequestError );
    requestor.load( request );
}

private function xmlRPCRequestComplete( event:Event ):void
{
    trace( XML(event.target.data).toXMLString() );
}

private function xmlRPCRequestError( error:ErrorEvent ):void
{
    trace( "An error occurred: " + error );
}
```

Da WebKit in AIR keine Unterstützung für die E4X-Syntax bietet, funktioniert die Methode, die im vorigen Beispiel zur Erstellung des XML-Dokuments verwendet wurde, in JavaScript-Code nicht. Stattdessen müssen Sie das XML-Dokument mit den DOM-Methoden erstellen. Oder Sie erstellen das Dokument als String und verwenden dann die DOMParser-Klasse in JavaScript, um den String in XML zu konvertieren.

Das folgende Beispiel verwendet DOM-Methoden zum Erstellen einer XML-RPC-Nachricht und eine XMLHttpRequest-Anforderung zum Durchführen der Webdiensttransaktion:

```
<html>
<head>
<title>XML-RPC web service request</title>
<script type="text/javascript">

function makeRequest()
{
    var requestDisplay = document.getElementById( "request" );
    var resultDisplay  = document.getElementById( "result" );

    var request = {};
    request.URL = "http://services.example.com/xmlrpc/";
    request.method = "test.echo";
    request.HTTPmethod = "POST";
    request.parameters = {};
    request.parameters.api_key = "123456ABC";
    request.parameters.message = "Able was I ere I saw Elba.";
    var requestMessage = formatXMLRPC( request );

    xmlhttp = new XMLHttpRequest();
    xmlhttp.open( request.HTTPmethod, request.URL, true);
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            resultDisplay.innerText = xmlhttp.responseText;
        }
    }
    xmlhttp.send( requestMessage );
```

```
    requestDisplay.innerText = xmlToString( requestMessage.documentElement );
}

//Formats a request as XML-RPC document
function formatXMLRPC( request )
{
    var xmldoc = document.implementation.createDocument( "", "", null );
    var root = xmldoc.createElement( "methodCall" );
    xmldoc.appendChild( root );
    var methodName = xmldoc.createElement( "methodName" );
    var methodString = xmldoc.createTextNode( request.method );
    methodName.appendChild( methodString );

    root.appendChild( methodName );

        var params = xmldoc.createElement( "params" );
        root.appendChild( params );

        var param = xmldoc.createElement( "param" );
        params.appendChild( param );
        var value = xmldoc.createElement( "value" );
        param.appendChild( value );
        var struct = xmldoc.createElement( "struct" );
        value.appendChild( struct );

        for( var property in request.parameters )
        {
            var member = xmldoc.createElement( "member" );
            struct.appendChild( member );

            var name = xmldoc.createElement( "name" );
            var paramName = xmldoc.createTextNode( property );
            name.appendChild( paramName )
            member.appendChild( name );

            var value = xmldoc.createElement( "value" );
            var type = xmldoc.createElement( "string" );
            value.appendChild( type );
            var paramValue = xmldoc.createTextNode( request.parameters[property] );
            type.appendChild( paramValue )
            member.appendChild( value );
        }
    return xmldoc;
}

//Returns a string representation of an XML node
function xmlToString( rootNode, indent )
{
    if( indent == null ) indent = "";
    var result = indent + "<" + rootNode.tagName + ">\n";
    for( var i = 0; i < rootNode.childNodes.length; i++)
    {
        if(rootNode.childNodes.item( i ).nodeType == Node.TEXT_NODE )
        {
            result += indent + "    " + rootNode.childNodes.item( i ).textContent + "\n";
        }
    }
```

```
    if( rootNode.childElementCount > 0 )
    {
        result += xmlToString( rootNode.firstElementChild, indent + "    " );
    }
    if( rootNode.nextElementSibling )
    {
        result += indent + "</" + rootNode.tagName + ">\n";
        result += xmlToString( rootNode.nextElementSibling, indent );
    }
    else
    {
        result += indent +"</" + rootNode.tagName + ">\n";
    }
    return result;
}

</script>
</head>
<body onload="makeRequest()">
<h1>Request:</h1>
<pre id="request"></pre>
<h1>Result:</h1>
<pre id="result"></pre>
</body>
</html>
```

SOAP-Webdienstanforderungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

SOAP baut auf dem allgemeinen Konzept des XML-RPC-Webdienstes auf und bietet eine vielseitigere, aber auch komplexere Technik zum Übertragen von typisierten Daten. SOAP-Webdienste enthalten normalerweise eine WSDL-Datei (Web Service Description Language), die die Aufrufe des Webdienstes, die Datentypen und die Dienst-URL angibt. ActionScript 3 bietet zwar keine direkte Unterstützung für SOAP, doch Sie können eine SOAP-XML-Nachricht „manuell“ erstellen, an den Server senden und dann die Ergebnisse analysieren. Sofern es sich nicht um einen äußerst einfachen SOAP-Webdienst handelt, ist dieses Verfahren jedoch äußerst zeitintensiv. Sie können viel Entwicklungszeit sparen, indem Sie stattdessen eine vorhandene SOAP-Bibliothek verwenden.

Das Flex-Framework umfasst Bibliotheken für den Zugriff auf SOAP-Webdienste. In Flash Builder ist die Bibliothek „rpc.swc“ automatisch in Flex-Projekten vorhanden, da sie zum Flex-Framework gehört. In Flash Professional können Sie die Flex-Bibliotheken „framework.swc“ und „rpc.swc“ dem Bibliothekspfad eines Projekts hinzufügen und dann mit ActionScript auf die Flex-Klassen zugreifen.

Verwandte Hilfethemen

Verwenden der Flex-Webdienstkomponente in Flash Professional

Cristophe Coenraets: Real-time Trader Desktop for Android

Öffnen einer URL in einer anderen Anwendung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `navigateToURL()`-Funktion können Sie eine URL in einem Webbrowser oder in einer anderen Anwendung öffnen. Für Inhalt, der in AIR ausgeführt wird, öffnet die `navigateToURL()`-Funktion die Seite im Standardwebbrowser des Systems.

Für das URLRequest-Objekt, das Sie als `request`-Parameter dieser Funktion übergeben, wird nur die `url`-Eigenschaft verwendet.

Beim ersten Parameter der `navigateToURL()`-Funktion, dem `navigate`-Parameter, handelt es sich um ein URLRequest-Objekt (siehe „Verwenden der URLRequest-Klasse“ auf Seite 865). Beim zweiten handelt es sich um einen optionalen `window`-Parameter, in dem Sie den Fensternamen angeben können. Mit dem folgenden Code wird zum Beispiel die Webseite www.adobe.com geöffnet:

```
var url:String = "http://www.adobe.com";
var urlReq:URLRequest = new URLRequest(url);
navigateToURL(urlReq);
```

Hinweis: *Beim Verwenden der `navigateToURL()`-Funktion behandelt die Laufzeitumgebung ein URLRequest-Objekt, das die POST-Methode verwendet (ein Objekt, dessen `method`-Eigenschaft auf `URLRequestMethod.POST` eingestellt ist), wie beim Verwenden der GET-Methode.*

Bei Verwendung der `navigateToURL()`-Funktion sind URI-Schemas abhängig von der Sicherheits-Sandbox des Codes, der die `navigateToURL()`-Funktion aufruft, zulässig.

Einige APIs ermöglichen es, Inhalt in einem Webbrowser zu starten. Aus Sicherheitsgründen sind einige URI-Schemas bei der Verwendung dieser APIs in AIR nicht zulässig. Welche Schemas unzulässig sind, hängt von der Sicherheits-Sandbox des Codes ab, der die API verwendet. (Weitere Informationen zu Sicherheits-Sandboxes finden Sie unter „AIR-Sicherheit“ auf Seite 1153.)

Anwendungs-Sandbox (nur AIR)

Für URLs, die von Inhalt in der AIR-Anwendungs-Sandbox aufgerufen werden, kann jedes URI-Schema verwendet werden. Eine Anwendung muss für die Verarbeitung des URI-Schemas registriert sein, andernfalls wird die Anforderung nicht ausgeführt. Die folgenden Schemas werden auf vielen Computern und Geräten unterstützt:

- `http:`
- `https:`
- `file:`
- `mailto:` – AIR leitet diese Anforderungen an die registrierte Mailanwendung des Systems weiter
- `sms:` – AIR leitet `sms:`-Anforderungen an die standardmäßige SMS-Anwendung weiter. Das Format der URL muss den Konventionen des Systems entsprechen, auf dem die Anwendung ausgeführt wird. Beispielsweise sieht das URI-Schema unter Android Kleinbuchstaben vor.

  ```
  navigateToURL( new URLRequest( "sms:+15555550101") );
  ```

- `tel:` – AIR leitet `tel:`-Anforderungen an die standardmäßige Telefonwahlanwendung weiter. Das Format der URL muss den Konventionen des Systems entsprechen, auf dem die Anwendung ausgeführt wird. Beispielsweise sieht das URI-Schema unter Android Kleinbuchstaben vor.

  ```
  navigateToURL( new URLRequest( "tel:5555555555") );
  ```

- `market:` – AIR leitet `market:`-Anforderungen an die Market-App weiter, die auf Android-Geräten normalerweise unterstützt wird.

```
navigateToURL( new URLRequest( "market://search?q=Adobe Flash") );
navigateToURL( new URLRequest( "market://search?q=pname:com.adobe.flashplayer") );
```

Wenn das Betriebssystem dies zulässt, können Anwendungen benutzerdefinierte URI-Schemas definieren und registrieren. Sie können eine URL mit dem Schema erstellen, um die Anwendung aus AIR zu starten.

Remote-Sandboxen

Die folgenden Schemas sind zulässig. Verwenden Sie diese Schemas genauso wie in einem Webbrowser.

- `http:`
- `https:`
- `mailto:` – AIR leitet diese Anforderungen an die registrierte Mailanwendung des Systems weiter

Alle anderen URI-Schemas sind unzulässig.

Lokale Sandbox des Dateisystems

Die folgenden Schemas sind zulässig. Verwenden Sie diese Schemas genauso wie in einem Webbrowser.

- `file:`
- `mailto:` – AIR leitet diese Anforderungen an die registrierte Mailanwendung des Systems weiter

Alle anderen URI-Schemas sind unzulässig.

Lokale Sandbox mit Netzwerkzugang

Die folgenden Schemas sind zulässig. Verwenden Sie diese Schemas genauso wie in einem Webbrowser.

- `http:`
- `https:`
- `mailto:` – AIR leitet diese Anforderungen an die registrierte Mailanwendung des Systems weiter

Alle anderen URI-Schemas sind unzulässig.

Lokale vertrauenswürdige Sandbox

Die folgenden Schemas sind zulässig. Verwenden Sie diese Schemas genauso wie in einem Webbrowser.

- `file:`
- `http:`
- https:
- `mailto:` – AIR leitet diese Anforderungen an die registrierte Mailanwendung des Systems weiter

Alle anderen URI-Schemas sind unzulässig.

Kapitel 45: Kommunikation mit anderen Flash Player- und AIR-Instanzen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die LocalConnection-Klasse ermöglicht die Kommunikation zwischen Adobe® AIR®-Anwendungen sowie zwischen SWF-Inhalt, der im Browser ausgeführt wird. Mit der LocalConnection-Klassen können Sie auch zwischen einer AIR-Anwendung und im Browser ausgeführtem SWF-Inhalt kommunizieren. Mit der LocalConnection-Klasse können Sie vielseitige Anwendungen erstellen, die Daten mit Flash Player- und AIR-Instanzen gemeinsam nutzen können.

LocalConnection-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der LocalConnection-Klasse können Sie SWF-Dateien entwickeln, die ohne die fscommand()-Methode oder JavaScript Anweisungen untereinander austauschen können. LocalConnection-Objekte ermöglichen die Kommunikation zwischen verschiedenen SWF-Dateien auf demselben Client-Computer, die auch in verschiedenen Anwendungen ausgeführt werden können. Beispielsweise können eine in einem Browser ausgeführte SWF-Datei und eine in einem Projektor ausgeführte SWF-Datei Daten gemeinsam nutzen. Dabei verwaltet der Projektor die lokalen Informationen und die browserbasierte SWF-Datei stellt eine Remote-Verbindung her. (Ein „Projektor“ ist eine SWF-Datei, die in einem Format gespeichert wurde, das als eigenständige Anwendung ausgeführt werden kann, d. h. Flash Player ist in die ausführbare Datei eingebettet und muss nicht installiert sein.)

LocalConnection-Objekte können zur Kommunikation zwischen SWF-Dateien verwendet werden, die mit verschiedenen ActionScript-Versionen erstellt wurden:

- In ActionScript 3.0 erstellte LocalConnection-Objekte können mit LocalConnection-Objekten kommunizieren, die in ActionScript 1.0 oder 2.0 erstellt wurden.
- In ActionScript 1.0 oder 2.0 erstellte LocalConnection-Objekte können mit LocalConnection-Objekten kommunizieren, die in ActionScript 3.0 erstellt wurden.

Flash Player verarbeitet die Kommunikation zwischen LocalConnection-Objekten verschiedener Versionen automatisch.

Die einfachste Art der Verwendung von LocalConnection-Objekten besteht darin, die Kommunikation nur zwischen LocalConnection-Objekten zuzulassen, die sich in derselben Domäne oder derselben AIR-Anwendung befinden. Dadurch brauchen Sie sich nicht um Sicherheitsfragen zu kümmern. Wenn die Kommunikation zwischen verschiedenen Domänen erforderlich ist, gibt es eine Reihe von Möglichkeiten, Sicherheitsmaßnahmen zu integrieren. Weitere Informationen finden Sie in der Beschreibung des `connectionName`-Parameters der `send()`-Methode sowie in den Einträgen zu `allowDomain()` und `domain` der LocalConnection-Klasse im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Sie können LocalConnection-Objekte zum Senden und Empfangen von Daten innerhalb einer SWF-Datei verwenden. Dies wird jedoch nicht empfohlen. Verwenden Sie stattdessen gemeinsame Objekte.

Es gibt drei Verfahren, Ihre LocalConnection-Objekte in Rückrufmethoden einzufügen:

- Erstellen einer Unterklasse der LocalConnection-Klasse und Hinzufügen von Methoden.

- Einstellen der `LocalConnection.client`-Eigenschaft auf ein Objekt, das die Methoden implementiert.
- Erstellen einer dynamischen Klasse, die LocalConnection erweitert, und dynamisches Anhängen der Methoden.

Das erste Verfahren, Rückrufmethoden hinzuzufügen, ist das Erweitern der LocalConnection-Klasse. Sie definieren die Methoden mit der benutzerdefinierten Klasse, anstatt sie der LocalConnection-Instanz dynamisch hinzuzufügen. Dieser Ansatz wird im folgenden Code gezeigt:

```
package
{
    import flash.net.LocalConnection;
    public class CustomLocalConnection extends LocalConnection
    {
        public function CustomLocalConnection(connectionName:String)
        {
            try
            {
                connect(connectionName);
            }
            catch (error:ArgumentError)
            {
                // server already created/connected
            }
        }
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

Um eine neue Instanz der CustomLocalConnection-Klasse zu erstellen, können Sie den folgenden Code verwenden:

```
var serverLC:CustomLocalConnection;
serverLC = new CustomLocalConnection("serverName");
```

Die zweite Möglichkeit, Rückrufmethoden hinzuzufügen, ist das Verwenden der Eigenschaft `LocalConnection.client`. Hierzu gehört das Erstellen einer benutzerdefinierten Klasse und das Zuweisen einer neuen Instanz zur `client`-Eigenschaft. Dies wird im folgenden Code gezeigt:

```
var lc:LocalConnection = new LocalConnection();
lc.client = new CustomClient();
```

Die Eigenschaft `LocalConnection.client` gibt die Objekt-Rückrufmethoden an, die aufgerufen werden müssen. Im vorangegangenen Code wurde die Eigenschaft `client` auf eine neue Instanz einer benutzerdefinierten Klasse eingestellt: CustomClient. Der Standardwert der Eigenschaft `client` ist die aktuelle LocalConnection-Instanz. Sie können die Eigenschaft `client` verwenden, wenn zwei Datenprozeduren den gleichen Methodensatz aufweisen, jedoch unterschiedlich agieren, beispielsweise in einer Anwendung, in der eine Schaltfläche in einem Fenster die Ansicht in einem zweiten Fenster umschaltet.

Zum Erstellen der CustomClient-Klasse können Sie den folgenden Code verwenden:

```
package
{
    public class CustomClient extends Object
    {
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

Die dritte Möglichkeit zum Hinzufügen von Rückrufmethoden, das Erstellen einer dynamischen Klasse und das dynamische Anhängen der Methoden, ähnelt dem Verwenden der LocalConnection-Klasse in früheren Versionen von ActionScript. Dies wird im folgenden Code gezeigt:

```
import flash.net.LocalConnection;
dynamic class DynamicLocalConnection extends LocalConnection {}
```

Rückrufmethoden können dieser Klasse mithilfe des folgenden Codes dynamisch hinzugefügt werden:

```
var connection:DynamicLocalConnection = new DynamicLocalConnection();
connection.onMethod = this.onMethod;
// Add your code here.
public function onMethod(timeString:String):void
{
    trace("onMethod called at: " + timeString);
}
```

Das zuvor beschriebene Verfahren zum Hinzufügen von Rückrufmethoden wird nicht empfohlen, da es dem Code an Portabilität mangelt. Darüber hinaus führt dieses Verfahren zum Erstellen lokaler Verbindungen möglicherweise zu Leistungsbeeinträchtigungen, da der Zugriff auf dynamische Eigenschaften deutlich langsamer erfolgt als der Zugriff auf versiegelte Eigenschaften.

isPerUser-Eigenschaft

Die `isPerUser`-Eigenschaft wurde in Flash Player (10.0.32) und AIR (1.5.2) eingeführt, um einen Konflikt zu beheben, der auftritt, wenn mehr als ein Benutzer bei einem Mac angemeldet ist. Unter anderen Betriebssystemen wird die Eigenschaft ignoriert, da die lokale Verbindung immer für einzelne Benutzer vorgesehen ist. Die `isPerUser`-Eigenschaft sollte in neuem Code auf `true` eingestellt werden. Zur Gewährleistung der Abwärtskompatibilität lautet der Standardwert derzeit jedoch `false`. Der Standardwert wird in zukünftigen Versionen der Laufzeiten möglicherweise geändert.

Senden von Meldungen zwischen zwei Anwendungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die LocalConnection-Klasse ermöglicht die Kommunikation zwischen verschiedenen AIR-Anwendungen sowie zwischen verschiedenen Adobe® Flash® Player (SWF)-Anwendungen, die in einem Browser ausgeführt werden. Mit der LocalConnection-Klassen können Sie auch zwischen einer AIR-Anwendung und einer im Browser ausgeführtem SWF-Anwendung kommunizieren.

Beispielsweise können Sie mehrere Flash Player-Instanzen auf einer Webseite integrieren oder eine Flash Player-Instanz Daten von einer zweiten Flash Player-Instanz in einem Popupfenster abrufen lassen.

Im folgenden Beispiel wird ein LocalConnection-Objekt definiert, das als Server fungiert und die von anderen Anwendungen eingehenden LocalConnection-Aufrufe akzeptiert:

```
package
{
    import flash.net.LocalConnection;
    import flash.display.Sprite;
    public class ServerLC extends Sprite
    {
        public function ServerLC()
        {
            var lc:LocalConnection = new LocalConnection();
            lc.client = new CustomClient1();
            try
            {
                lc.connect("conn1");
            }
            catch (error:Error)
            {
                trace("error:: already connected");
            }
        }
    }
}
```

Zuerst wird ein LocalConnection-Objekt namens `lc` erstellt und anschließend wird für die `client`-Eigenschaft ein `clientObject`-Objekt festgelegt. Ruft eine andere Anwendung in dieser LocalConnection-Instanz eine Methode auf, sucht die Laufzeit im `clientObject`-Objekt nach dieser Methode.

Wenn Sie bereits über eine Verbindung mit dem angegebenen Namen verfügen, wird eine Argument-Fehlerausnahme ausgelöst, die darauf hinweist, dass der Versuch zur Verbindungsherstellung fehlgeschlagen ist, da das Objekt bereits verbunden ist.

Immer wenn eine Flash Player-Instanz eine Verbindung mit dieser SWF-Datei herstellt und versucht, eine Methode für die angegebene lokale Verbindung aufzurufen, wird die Anforderung an die von der `client`-Eigenschaft angegebene Klasse gesendet. Diese Eigenschaft ist auf die CustomClient1-Klasse eingestellt:

```
package
{
    import flash.events.*;
    import flash.system.fscommand;
    import flash.utils.Timer;
    public class CustomClient1 extends Object
    {
        public function doMessage(value:String = ""):void
        {
            trace(value);
        }
        public function doQuit():void
        {
            trace("quitting in 5 seconds");
            this.close();
            var quitTimer:Timer = new Timer(5000, 1);
            quitTimer.addEventListener(TimerEvent.TIMER, closeHandler);
        }
        public function closeHandler(event:TimerEvent):void
        {
            fscommand("quit");
        }
    }
}
```

Zum Erstellen eines LocalConnection-Servers rufen Sie die `LocalConnection.connect()`-Methode auf und geben einen einmaligen Verbindungsnamen ein. Wenn bereits eine Verbindung mit dem angegebenen Namen besteht, wird ein ArgumentError-Fehler erzeugt, der darauf hinweist, dass die versuchte Verbindung fehlgeschlagen ist, da das Objekt bereits eine Verbindung hergestellt hat.

Im folgenden Codebeispiel wird gezeigt, wie ein LocalConnection-Objekt namens `conn1` erstellt wird:

```
try
{
    connection.connect("conn1");
}
catch (error:ArgumentError)
{
    trace("Error! Server already exists\n");
}
```

Um eine Verbindung zwischen der primären und sekundären Anwendung herzustellen, müssen Sie zuerst im sendenden LocalConnection-Objekt ein LocalConnection-Objekt erstellen. Rufen Sie dann die `LocalConnection.send()`-Methode mit dem Namen der Verbindung und dem Namen der auszuführenden Methode auf. Um beispielsweise die `doQuit`-Methode an das zuvor erstellte LocalConnection-Objekt zu senden, verwenden Sie den folgenden Code:

```
sendingConnection.send("conn1", "doQuit");
```

In diesem Beispiel wird eine Verbindung zu einem vorhandenen LocalConnection-Objekt mit dem Verbindungsnamen `conn1` hergestellt und die `doMessage()`-Methode in der Remote-Anwendung aufgerufen. Um Parameter an die Remote-Anwendung zu senden, geben Sie in der `send()`-Methode nach dem Methodennamen zusätzliche Argumente an (siehe folgender Ausschnitt):

```
sendingConnection.send("conn1", "doMessage", "Hello world");
```

Herstellen von Verbindungen mit Inhalten in verschiedenen Domänen und mit AIR-Anwendungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um nur die Kommunikation von bestimmten Domänen zuzulassen, rufen Sie die `allowDomain()`- oder `allowInsecureDomain()`-Methode der LocalConnection-Klasse auf und übergeben eine Liste mit der Domäne oder den Domänen, der bzw. denen der Zugriff auf das LocalConnection-Objekt erlaubt ist, und übergeben den oder die Namen der zulässigen Domänen.

In früheren Versionen von ActionScript waren `LocalConnection.allowDomain()` und `LocalConnection.allowInsecureDomain()` Rückrufmethoden, die von den Entwicklern implementiert werden mussten und einen booleschen Wert zurückgaben. In ActionScript 3.0 sind `LocalConnection.allowDomain()` und `LocalConnection.allowInsecureDomain()` integrierte Methoden, die Entwickler genauso wie `Security.allowDomain()` und `Security.allowInsecureDomain()` aufrufen können, um mindestens einen Domänennamen zu übergeben, für den der Zugriff erlaubt ist.

In Flash Player 8 wurden Sicherheitsbeschränkungen für lokale SWF-Dateien eingeführt. SWF-Dateien, die auf das Internet zugreifen können, haben keinen Zugriff auf das lokale Dateisystem. Wenn Sie `localhost` angeben, können alle lokalen SWF-Dateien auf diese SWF-Datei zugreifen. Wenn die `LocalConnection.send()`-Methode versucht, mit einer SWF-Datei in einer Sicherheits-Sandbox zu kommunizieren, auf die der aufrufende Code keinen Zugriff hat, wird ein `securityError`-Ereignis(`(ecurityErrorEvent.SECURITY_ERROR`) ausgelöst. Zur Umgehung dieses Fehlers können Sie in der `LocalConnection.allowDomain()`-Methode des Empfängers die Domäne des Aufrufers angeben.

Es gibt zwei spezielle Werte, die an die `LocalConnection.allowDomain()`- und `LocalConnection.allowInsecureDomain()`-Methoden übergeben werden können: `*` und `localhost`. Der Sternwert `(*)` ermöglicht den Zugriff von allen Domänen. Der String `localhost` ermöglicht es Inhalt, der zwar lokal aber außerhalb des Ressourcenverzeichnisses der Anwendung installiert ist, die Anwendung aufzurufen.

Wenn versucht wird, mithilfe der `LocalConnection.send()`-Methode mit einer Anwendung in einer Sicherheits-Sandbox zu kommunizieren, auf die der aufrufende Code keinen Zugriff hat, wird ein `securityError`-Ereignis(`SecurityErrorEvent.SECURITY_ERROR`) ausgelöst. Zur Umgehung dieses Fehlers können Sie in der `LocalConnection.allowDomain()`-Methode des Empfängers die Domäne des Aufrufers angeben.

Wenn Sie die Kommunikation nur zwischen Inhalt in derselben Domäne implementieren, können Sie einen `connectionName`-Parameter angeben, der nicht mit einem Unterstrich (_) beginnt und der keinen Domänennamen angibt (beispielsweise `myDomain:connectionName`). Verwenden Sie denselben String im Befehl `LocalConnection.connect(verbindungsName)`.

Wenn Sie die Kommunikation nur zwischen Inhalten in unterschiedlichen Domänen implementieren, geben Sie einen `connectionName` -Parameter an, der mit einem Unterstrich beginnt. Dadurch wird die Portierbarkeit des Inhalts mit dem empfangenden LocalConnection-Objekt zwischen Domänen erhöht. Im Folgenden werden zwei mögliche Fälle erläutert:

- Wenn der String für `connectionName` nicht mit einem Unterstrich (_) beginnt, fügt die Laufzeitumgebung ein Präfix mit dem Namen der Superdomäne und einen Doppelpunkt hinzu (z. B. `myDomain:connectionName`). Dies stellt zwar sicher, dass es keinen Konflikt zwischen Ihrer Verbindung und gleichnamigen Verbindungen aus anderen Domänen gibt, allerdings müssen sendende LocalConnection-Objekte ebenfalls diese Superdomäne angeben (z. B. `myDomain:connectionName`). Wenn die HTML- oder SWF-Datei mit dem empfangenden LocalConnection-Objekt in eine andere Domäne verschoben wird, ändert die Laufzeitumgebung das Präfix entsprechend der neuen Superdomäne (z. B. `anotherDomain:connectionName`). Alle sendenden LocalConnection-Objekte müssen dann manuell bearbeitet werden, damit sie auf die neue Superdomäne verweisen.
- Wenn der String für `connectionName` mit einem Unterstrich beginnt (z. B. `_connectionName`), fügt die Laufzeitumgebung dem String kein Präfix hinzu. Das heißt, empfangende und sendende LocalConnection-Objekte verwenden identische Strings für `connectionName`. Wenn das empfangende Objekt mit `LocalConnection.allowDomain()` angibt, dass Verbindungen von allen Domänen zulässig sind, können Sie die HTML- oder SWF-Datei mit dem empfangenden LocalConnection-Objekt in eine andere Domäne verschieben, ohne die sendenden LocalConnection-Objekte zu ändern.

  Ein Nachteil, den die Verwendung von Namen mit Unterstrich in `connectionName` mit sich bringt, sind die möglichen Konflikte: wenn z. B. zwei Anwendungen versuchen, eine Verbindung mit demselben `connectionName` herzustellen. Im Zusammenhang damit gibt es einen zweiten, die Sicherheit betreffenden Nachteil. Verbindungsnamen, die die Unterstrichsyntax verwenden, identifizieren nicht die Domäne der wartenden Anwendung. Daher werden domänenqualifizierte Namen bevorzugt.

Adobe AIR

Zur Kommunikation mit Inhalt, der in der Sicherheits-Sandbox der AIR-Anwendung ausgeführt wird (mit der AIR-Anwendung installierter Inhalt), müssen Sie dem Verbindungsnamen als Präfix eine Superdomäne voranstellen, die die AIR-Anwendung identifiziert. Der String für die Superdomäne beginnt mit `app#`, gefolgt von der Anwendungs-ID und einem Punkt (.), gefolgt von der Herausgeber-ID (sofern definiert). Die richtige Superdomäne für den `connectionName`-Parameter einer Anwendung mit der Anwendungs-ID `com.example.air.MyApp` und ohne Herausgeber-ID lautet beispielsweise `"app#com.example.air.MyApp"`. Wenn der Basisverbindungsname beispielsweise „appConnection“ lautet, muss für den `connectionName`-Parameter der folgende String verwendet werden: `"app#com.example.air.MyApp:appConnection"`. Wenn die Anwendung eine Herausgeber-ID enthält, muss auch diese ID in den String für die Superdomäne aufgenommen werden: `"app#com.example.air.MyApp.B146A943FBD637B68C334022D304CEA226D129B4.1"`.

Wenn Sie zulassen, dass eine andere AIR-Anwendung mit Ihrer Anwendung über die lokale Verbindung kommuniziert, müssen Sie die `allowDomain()`-Methode des LocalConnection-Objekts aufrufen und den Domänennamen der lokalen Verbindung übergeben. Bei einer AIR-Anwendung wird dieser Domänenname aus den Anwendungs- und Herausgeber-IDs in gleicher Weise wie der Verbindungsstring gebildet. Beispiel: Wenn die Anwendungs-ID der sendenden AIR-Anwendung `com.example.air.FriendlyApp` und die Herausgeber-ID `214649436BD677B62C33D02233043EA236D13934.1` lautet, dann lautet der Domänenstring, mit dem Sie die Verbindung dieser Anwendung zulassen, `app#com.example.air.FriendlyApp.214649436BD677B62C33D02233043EA236D13934.1`. (Ab AIR 1.5.3 verfügen nicht alle AIR-Anwendungen über Herausgeber-IDs.)

Kapitel 46: Kommunikation mit nativen Prozessen in AIR

Adobe AIR 2 und höher

Ab Adobe AIR 2 können AIR-Anwendungen über die Befehlszeile ausgeführt werden und mit anderen nativen Prozessen kommunizieren. Beispielsweise kann eine AIR-Anwendung über die standardmäßigen Ein- und Ausgabestreams einen Prozess ausführen und mit ihm kommunizieren.

Zur Kommunikation mit nativen Prozessen muss eine AIR-Anwendung so verpackt werden, dass sie über ein natives Installationsprogramm installiert wird. Der Dateityp eines nativen Installationsprogramms richtet sich nach dem jeweiligen Betriebssystem:

- DMG-Datei unter Mac OS.
- EXE-Datei unter Windows.
- RPM- oder DEB-Paket unter Linux.

Diese Anwendungen werden als erweiterte Desktop-Profilanwendungen bezeichnet. Sie erstellen eine native Installationsdatei, indem Sie die `-target native`-Option angeben, wenn Sie den `-package`-Befehl mit ADT aufrufen.

Verwandte Hilfethemen

flash.filesystem.File.openWithDefaultApplication()

flash.desktop.NativeProcess

Überblick über die Kommunikation mit nativen Prozessen

Adobe AIR 2 und höher

Eine AIR-Anwendung in einem erweiterten Desktop-Profil kann eine Datei genauso ausführen wie beim Aufruf über die Befehlszeile. Sie kann mit den Standardstreams des nativen Prozesses kommunizieren. Zu den Standardstreams zählen der Standardeingabestream (stdin), der Ausgabestream (stdout) und der Standardfehlerstream (stderr).

Hinweis: *Anwendungen im erweiterten Desktop-Profil können Dateien und Anwendungen auch über die `File.openWithDefaultApplication()`-Methode starten. Bei Verwendung dieser Methode hat die AIR-Anwendung jedoch keinen Zugriff auf die Standardstreams. Weitere Informationen finden Sie unter „Öffnen von Dateien mit der Standardsystemanwendung“ auf Seite 726.*

Das folgende Codebeispiel zeigt, wie die Anwendung „test.exe“ im Anwendungsverzeichnis gestartet wird. Die Anwendung übergibt `"hello"` als Befehlszeilenargument und fügt dem Standardausgabestream des Prozesses einen Ereignis-Listener hinzu:

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
var processArgs:Vector.<String> = new Vector.<String>();
processArgs.push("hello");
nativeProcessStartupInfo.arguments = processArgs;
process = new NativeProcess();
process.addEventListener(ProgressEvent.STANDARD_OUTPUT_DATA, onOutputData);
process.start(nativeProcessStartupInfo);
public function onOutputData(event:ProgressEvent):void
{
    var stdOut:ByteArray = process.standardOutput;
    var data:String = stdOut.readUTFBytes(process.standardOutput.bytesAvailable);
    trace("Got: ", data);
}
```

Starten und Schließen eines nativen Prozesses

Adobe AIR 2 und höher

Zum Starten eines nativen Prozesses richten Sie ein NativeProcessInfo-Objekt ein, das folgende Schritte ausführt:

- Verweisen auf die zu startende Datei
- Speichern von Befehlszeilenargumenten, die nach dem Starten an den Prozess übergeben werden sollen (optional)
- Festlegen des Arbeitsverzeichnisses für den Prozess (optional)

Zum Starten des nativen Prozesses muss das NativeProcessInfo-Objekt als Parameter der `start()`-Methode eines NativeProcess-Objekts übergeben werden.

Der folgende Code zeigt beispielsweise, wie die Anwendung „test.exe“ im Anwendungsverzeichnis gestartet wird. Die Anwendung übergibt das Argument `"hello"` und legt das Dokumentverzeichnis des Anwenders als Arbeitsverzeichnis fest:

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
var processArgs:Vector.<String> = new Vector.<String>();
processArgs[0] = "hello";
nativeProcessStartupInfo.arguments = processArgs;
nativeProcessStartupInfo.workingDirectory = File.documentsDirectory;
process = new NativeProcess();
process.start(nativeProcessStartupInfo);
```

Zum Beenden des Prozesses rufen Sie die `exit()`-Methode des NativeProcess-Objekts auf.

Wenn eine Datei in der installierten Anwendung ausführbar sein soll, muss sie im Dateisystem ausführbar sein, wenn Sie die Anwendung verpacken. (Unter Mac und Linux können Sie das Flag für eine ausführbare Datei bei Bedarf mithilfe des chmod-Befehls festlegen.)

Kommunikation mit einem nativen Prozess

Adobe AIR 2 und höher

Nachdem eine AIR-Anwendung einen nativen Prozess gestartet hat, ist die Kommunikation mit dem Standardeingabestream, dem Standardausgabestream und dem Standardfehlerstream des Prozesses möglich.

Daten werden mit den folgenden Eigenschaften des NativeProcess-Objekts aus den Streams gelesen und in die Streams geschrieben:

- `standardInput` – Ermöglicht den Zugriff auf die Daten des Standardeingabestreams.
- `standardOutput` – Ermöglicht den Zugriff auf die Daten des Standardausgabestreams.
- `standardError` – Ermöglicht den Zugriff auf die Daten des Standardfehlerstreams.

Schreiben in den Standardeingabestream

Zum Schreiben von Daten in den Standardeingabestream können Sie die write-Methoden der `standardInput`-Eigenschaft des NativeProcess-Objekts verwenden. Während die AIR-Anwendung Daten in den Prozess schreibt, setzt das NativeProcess-Objekt `standardInputProgress`-Ereignisse ab.

Wenn beim Schreiben in den Standardeingabestream ein Fehler auftritt, setzt das NativeProcess-Objekt ein `ioErrorStandardInput`-Ereignis ab.

Sie können den Eingabestream schließen, indem Sie die `closeInput()`-Methode des NativeProcess-Objekts aufrufen. Wenn der Eingabestream geschlossen wird, setzt das NativeProcess-Objekt ein `standardInputClose`-Ereignis ab.

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
process = new NativeProcess();
process.start(nativeProcessStartupInfo);
process.standardInput.writeUTF("foo");
if(process.running)
{
    process.closeInput();
}
```

Lesen aus dem Standardausgabestream

Zum Lesen von Daten aus dem Standardausgabestream verwenden Sie die read-Methoden dieser Eigenschaft. Während die AIR-Anwendung Ausgabestreamdaten vom Prozess erhält, setzt das NativeProcess-Objekt `standardOutputData`-Ereignisse ab.

Wenn beim Schreiben in den Standardausgabestream ein Fehler auftritt, setzt das NativeProcess-Objekt ein `standardOutputError`-Ereignis ab.

Wenn der Prozess den Ausgabestream schließt, setzt das NativeProcess-Objekt ein `standardOutputClose`-Ereignis ab.

Achten Sie beim Lesen von Daten aus dem Standardeingabestream darauf, die Daten direkt bei der Erstellung zu lesen. Fügen Sie also einen Ereignis-Listener für das `standardOutputData`-Ereignis hinzu. Im `standardOutputData`-Ereignis-Listener lesen Sie die Daten aus der `standardOutput`-Eigenschaft des NativeProcess-Objekts. Warten Sie nicht darauf, dass die Ereignisse `standardOutputClose` oder `exit` alle Daten lesen. Wenn Sie die Daten nicht direkt beim Erstellen durch den nativen Prozess lesen, kann der Puffer zu voll werden oder Daten können verloren gehen. Wenn der Puffer voll ist, kann dies dazu führen, dass der native Prozess beim Versuch, weitere Daten zu schreiben, angehalten wird. Wenn Sie jedoch keinen Ereignis-Listener für das `standardOutputData`-Ereignis registrieren, wird der Puffer nicht voll und der Prozess wird nicht angehalten. In diesem Fall haben Sie keinen Zugriff auf die Daten.

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
process = new NativeProcess();
process.addEventListener(ProgressEvent.STANDARD_OUTPUT_DATA, dataHandler);
process.start(nativeProcessStartupInfo);
var bytes:ByteArray = new ByteArray();
function dataHandler(event:ProgressEvent):void
{
    bytes.writeBytes(process.standardOutput.readBytes(process.standardOutput.bytesAvailable);
}
```

Lesen aus dem Standardfehlerstream

Zum Lesen von Daten aus dem Standardfehlerstream verwenden Sie die read-Methoden dieser Eigenschaft. Während die AIR-Anwendung Fehlerstreamdaten aus dem Prozess liest, setzt das NativeProcess-Objekt `standardErrorData`-Ereignisse ab.

Wenn beim Schreiben in den Standardfehlerstream ein Fehler auftritt, setzt das NativeProcess-Objekt ein `standardErrorIoError`-Ereignis ab.

Wenn der Prozess den Fehlerstream schließt, setzt das NativeProcess-Objekt ein `standardErrorClose`-Ereignis ab..

Achten Sie beim Lesen von Daten aus dem Standardfehlerstream darauf, die Daten direkt bei der Erstellung zu lesen. Fügen Sie also einen Ereignis-Listener für das `standardErrorData`-Ereignis hinzu. Im `standardErrorData`-Ereignis-Listener lesen Sie Daten aus der `standardError`-Eigenschaft des NativeProcess-Objekts. Warten Sie nicht darauf, dass die Ereignisse `standardErrorClose` oder `exit` alle Daten lesen. Wenn Sie die Daten nicht direkt beim Erstellen durch den nativen Prozess lesen, kann der Puffer zu voll werden oder Daten können verloren gehen. Wenn der Puffer voll ist, kann dies dazu führen, dass der native Prozess beim Versuch, weitere Daten zu schreiben, angehalten wird. Wenn Sie jedoch keinen Ereignis-Listener für das `standardErrorData`-Ereignis registrieren, wird der Puffer nicht voll und der Prozess wird nicht angehalten. In diesem Fall haben Sie keinen Zugriff auf die Daten.

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
process = new NativeProcess();
process.addEventListener(ProgressEvent.STANDARD_ERROR_DATA, errorDataHandler);
process.start(nativeProcessStartupInfo);
var errorBytes:ByteArray = new ByteArray();
function errorDataHandler(event:ProgressEvent):void
{
    bytes.writeBytes(process.standardError.readBytes(process.standardError.bytesAvailable);
}
```

Sicherheitsaspekte bei der Kommunikation mit nativen Prozessen

Adobe AIR 2 und höher

Die API für native Prozesse kann jede ausführbare Datei auf dem Benutzersystem ausführen. Seien Sie beim Konstruieren und Ausführen von Befehlen äußerst vorsichtig. Wenn ein Teil eines auszuführenden Befehls aus einer externen Quelle stammt, prüfen Sie sorgfältig, ob die Ausführung des Befehls sicher ist. Ebenso sollte die AIR-Anwendung alle Daten überprüfen, die an einen aktiven Prozess übergeben werden.

Die Validierung von Eingaben kann jedoch schwierig sein. Um diese Probleme zu vermeiden, schreiben Sie am besten eine native Anwendung (zum Beispiel eine EXE-Datei für Windows) mit spezifischen APIs. Diese APIs sollten nur die Befehle verarbeiten, die von der Anwendung definiert sind. Zum Beispiel könnte die Anwendung nur einen begrenzten Satz von Anweisungen über den Standardeingabestream akzeptieren.

AIR unter Windows lässt die direkte Ausführung von .bat-Dateien nicht zu. Die Befehlsinterpreter-Anwendung (cmd.exe) führt Windows-Dateien mit der Erweiterung „.bat“ aus. Wenn Sie eine .bat-Datei aufrufen, kann diese Befehlsanwendung Argumente, die an den Befehl übergeben werden, als zusätzlich zu startende Anwendungen interpretieren. Durch das böswillige Einfügen von zusätzlichen Zeichen in den Argumentstring könnte cmd.exe eine schädliche oder unsichere Anwendung ausführen. Ohne eine gute Datenvalidierung könnte Ihre Anwendung zum Beispiel `myBat.bat myArguments c:/evil.exe` aufrufen. Die Befehlsanwendung würde die Anwendung „evil.exe“ zusätzlich zu Ihrer Batchdatei ausführen.

Kapitel 47: Verwenden der externen API

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die externe API für ActionScript 3.0 (flash.external.ExternalInterface) ermöglicht eine einfache Kommunikation zwischen ActionScript und der Containeranwendung, in der Adobe Flash Player ausgeführt wird. Erstellen Sie mit der ExternalInterface-API eine Interaktion zwischen einem SWF-Dokument und JavaScript auf einer HTML-Seite.

Sie können die externe API verwenden, um mit einer Containeranwendung zu interagieren und Daten zwischen ActionScript und JavaScript auf einer HTML-Seite zu übergeben.

Häufige Aufgaben der externen API sind beispielsweise:

- Abrufen von Informationen über die Containeranwendung
- Verwenden von ActionScript, um Code auf einer Webseite aufzurufen, die in einem Browser oder einer AIR-Desktopanwendung angezeigt wird
- Aufrufen von ActionScript-Code von einer Webseite
- Erstellen einer Proxy-Klasse zum Vereinfachen des Aufrufs von ActionScript-Code von einer Webseite

Hinweis: *In dieser Beschreibung der externen Schnittstelle wird ausschließlich die Kommunikation zwischen ActionScript in einer SWF-Datei und der Containeranwendung beschrieben, die einen Verweis auf Flash Player oder die Instanz enthält, in der die SWF-Datei geladen wurde. Alle anderen Verwendungsmöglichkeiten von Flash Player innerhalb einer Anwendung werden in diesem Handbuch nicht behandelt. Flash Player ist zur Verwendung als Browser-Plug-In oder als Projektoranwendung (eigenständige Anwendung) vorgesehen. Andere Nutzungsszenarien werden möglicherweise nur begrenzt unterstützt.*

Verwenden der externen API in AIR

Da eine AIR-Anwendung keinen externen Container hat, ist diese externe Schnittstelle im Allgemeinen weder zutreffend noch erforderlich. Wenn Ihre AIR-Anwendung eine SWF-Datei direkt lädt, kann der Anwendungscode direkt mit dem ActionScript-Code in der SWF-Datei kommunizieren (sofern die Sicherheits-Sandbox keine Einschränkungen vorgibt).

Wenn die AIR-Anwendung jedoch eine SWF-Datei über eine HTML-Seite in einem HTMLLoader-Objekt lädt (oder eine HTML-Komponente in Flex), dient das HTMLLoader-Objekt als externer Container. Deshalb können Sie die externe Schnittstelle für die Kommunikation zwischen dem ActionScript-Code in der geladenen SWF-Datei und dem JavaScript-Code in der im HTMLLoader-Objekt geladenen HTML-Seite verwenden.

Grundlagen der Verwendung der externen API

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Obwohl SWF-Dateien in einigen Fällen eigenständig ausgeführt werden können (wenn Sie beispielsweise mit Adobe® Flash® Professional eine SWF-Projektoranwendung erstellen), werden SWF-Anwendungen in der Mehrzahl der Fälle als Elemente innerhalb anderer Anwendungen ausgeführt. In der Regel befindet sich die SWF-Datei in einem HTML-Datei-Container. Etwas weniger häufig werden SWF-Dateien für die gesamte oder für Teile der Benutzeroberfläche von Desktopanwendungen eingesetzt.

Beim Entwickeln fortgeschrittener Anwendungen ist es unter Umständen erforderlich, einen Datenaustausch zwischen der SWF-Datei und der Containeranwendung einzurichten. Es ist beispielsweise üblich, dass Text und andere Informationen auf einer Webseite als HTML angezeigt werden und dass für dynamische grafische Inhalte wie Diagramme oder Videos eine SWF-Datei eingefügt wird. In diesen Fällen ist es unter Umständen wünschenswert, dass eine Änderung in der SWF-Datei erfolgt, wenn der Benutzer auf eine Schaltfläche der Webseite klickt. ActionScript enthält einen Mechanismus (als „externe API" bezeichnet), der diese Art der Kommunikation zwischen ActionScript in einer SWF-Datei und anderem Code in der Containeranwendung ermöglicht.

Wichtige Konzepte und Begriffe

In der folgenden Referenzliste sind wichtige Begriffe aufgeführt, die mit dieser Funktion zu tun haben:

Containeranwendung Die Anwendung, in der Flash Player eine SWF-Datei ausführt, zum Beispiel ein Webbrowser und eine HTML-Seite mit Flash Player-Inhalt, oder eine AIR-Anwendung, die die SWF-Datei auf einer Webseite lädt.

Projektor Eine ausführbare Datei, die SWF-Inhalt sowie eine eingebettete Flash Player-Version enthält. Sie können einen Projektor mithilfe von Flash Professional oder des eigenständigen Flash Player erstellen. Sie werden im Allgemeinen eingesetzt, um SWF-Dateien auf CD-ROM bereitzustellen oder wenn die Downloadgröße keine Rolle spielt und der Autor der SWF-Datei sicherstellen möchte, dass diese auf jeden Fall ausgeführt werden kann, selbst wenn Flash Player auf dem Computer des Benutzers nicht installiert ist.

Proxy Eine Vermittlungsanwendung oder entsprechender Code, die Code in einer Anwendung (die „externe Anwendung") für eine andere Anwendung (die „aufrufende Anwendung") aufrufen und der aufrufenden Anwendung Werte zurückgeben. Ein Proxy kann aus unterschiedlichen Gründen verwendet werden, u. a.:

- Zum Vereinfachen des Aufrufens der externen Funktionen durch Konvertieren der entsprechenden Funktionsaufrufe der aufrufenden Anwendung in das von der externen Anwendung verwendete Format.
- Zum Umgehen von Sicherheits- und anderen Beschränkungen, die verhindern, dass die aufrufende Anwendung direkt mit der externen Anwendung kommuniziert.

Serialisieren Konvertieren von Objekten oder Datenwerten in ein Format, in dem die Werte in Nachrichten zwischen zwei Programmiersystemen übergeben werden können, beispielsweise über das Internet oder zwischen zwei unterschiedlichen Anwendungen, die auf demselben Computer ausgeführt werden.

Verwenden der Beispiele

Bei vielen Codebeispielen handelt es sich um kurze Codeausschnitte zu Demonstrationszwecken und nicht um voll einsatzbereite Beispiele oder um Code, mit dem Werte überprüft werden. Da bei Verwendung der externen API (per Definition) ActionScript-Code sowie Code in einer Containeranwendung geschrieben werden muss, wird beim Testen der Beispiele auch ein Container erstellt (z. B. eine Webseite mit der SWF-Datei). Darüber hinaus erfolgt mithilfe der Codebeispiele eine Interaktion mit dem Container.

So testen Sie ein Beispiel für die Kommunikation zwischen ActionScript und JavaScript:

1. Erstellen Sie mit Flash Professional ein neues Dokument und speichern Sie es auf dem Computer.
2. Wählen Sie im Hauptmenü die Optionen „Datei" > „Einstellungen für Veröffentlichungen" aus.
3. Überprüfen Sie im Dialogfeld „Einstellungen für Veröffentlichungen" auf der Registerkarte „Formate", ob die Kontrollkästchen „Flash" und „HTML" aktiviert sind.
4. Klicken Sie auf die Schaltfläche „Veröffentlichen". Dadurch werden eine SWF-Datei und eine HTML-Datei in dem Ordner und mit dem Namen erstellt, unter denen Sie auch das Dokument gespeichert haben. Klicken Sie auf „OK", um das Dialogfeld „Einstellungen für Veröffentlichungen" zu schließen.

5 Deaktivieren Sie das Kontrollkästchen „HTML“. Nachdem die HTML-Seite nun erstellt wurde, bearbeiten Sie sie, und fügen Sie den entsprechenden JavaScript-Code ein. Durch Deaktivieren des Kontrollkästchens „HTML“ wird sichergestellt, dass Ihre Änderungen an der HTML-Seite beim Veröffentlichen der SWF-Datei in Flash nicht mit einer neuen HTML-Seite überschrieben werden.

6 Klicken Sie auf „OK“, um das Dialogfeld „Einstellungen für Veröffentlichungen“ zu schließen.

7 Öffnen Sie die HTML-Datei, die in Flash beim Veröffentlichen der SWF-Datei erstellt wurde, mit einer HTML-Anwendung oder einem Texteditor. Fügen Sie im HTML-Quellcode öffnende und schließende `script`-Tags ein und kopieren Sie sie in den JavaScript-Code aus dem Codebeispiel:

```
<script>
// add the sample JavaScript code here
</script>
```

8 Speichern Sie die HTML-Datei und kehren Sie zu Flash zurück.

9 Wählen Sie das Schlüsselbild auf Bild 1 der Zeitleiste aus und öffnen Sie das Bedienfeld „Aktionen“.

10 Kopieren Sie das ActionScript-Codebeispiel in das Bedienfeld „Skript“.

11 Wählen Sie im Hauptmenü die Optionen „Datei“ > „Veröffentlichen“ aus, um die SWF-Datei mit den vorgenommenen Änderungen zu aktualisieren.

12 Öffnen Sie die bearbeitete HTML-Seite in einem Webbrowser, um die Seite anzuzeigen und die Kommunikation zwischen ActionScript und der HTML-Seite zu testen.

So testen Sie ein Beispiel für die Kommunikation zwischen ActionScript und einem ActiveX-Container:

1 Erstellen Sie mit Flash Professional ein neues Dokument und speichern Sie es auf dem Computer. Es empfiehlt sich, das Dokument in dem Ordner zu speichern, in dem die Containeranwendung normalerweise nach der SWF-Datei sucht.

2 Wählen Sie im Hauptmenü die Optionen „Datei“ > „Einstellungen für Veröffentlichungen“ aus.

3 Vergewissern Sie sich im Dialogfeld „Einstellungen für Veröffentlichungen“ auf der Registerkarte „Formate“, dass nur das Kontrollkästchen „Flash“ aktiviert ist.

4 Klicken Sie im Feld „Datei“ neben dem Kontrollkästchen „Flash“ auf das Ordnersymbol, um den Ordner auszuwählen, in dem die SWF-Datei veröffentlicht wird. Durch Festlegen des Speicherorts für die SWF-Datei können Sie (beispielsweise) das Dokument in einem Ordner und die veröffentlichte SWF-Datei in einem anderen Ordner speichern, z. B. in dem Ordner, in dem auch der Quellcode für die Containeranwendung gespeichert ist.

5 Wählen Sie das Schlüsselbild auf Bild 1 der Zeitleiste aus und öffnen Sie das Bedienfeld „Aktionen“.

6 Kopieren Sie das ActionScript-Codebeispiel in das Bedienfeld „Skript“.

7 Wählen Sie im Hauptmenü die Optionen „Datei“ > „Veröffentlichen“ aus, um die SWF-Datei erneut zu veröffentlichen.

8 Erstellen und führen Sie die Containeranwendung aus, um die Kommunikation zwischen ActionScript und der Containeranwendung zu testen.

Vollständige Beispiele für die Verwendung der externen API zur Kommunikation mit einer HTML-Seite finden Sie unter dem folgenden Thema:

- „Beispiel für externe API: Kommunikation zwischen ActionScript und JavaScript in einem Webbrowser“ auf Seite 904

Diese Beispiele enthalten den vollständigen Code, einschließlich ActionScript-Code und Code zur Fehlerüberprüfung in der Containeranwendung, den Sie beim Schreiben von Code für die externe API verwenden sollten. Ein weiteres vollständiges Beispiel zur Verwendung der externen API finden Sie im Beispiel für die ExternalInterface-Klasse in der ActionScript 3.0-Referenz.

Anforderungen und Vorteile externer APIs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die externe API ist der Teil von ActionScript, der einen Mechanismus für den Datenaustausch zwischen ActionScript und anderem Code bereitstellt, der in einer „externen Anwendung" ausgeführt wird, die als Container für Flash Player fungiert (gewöhnlich ein Webbrowser oder eine eigenständige Projektoranwendung). In ActionScript 3.0 wird die Funktionalität der externen API durch die ExternalInterface-Klasse bereitgestellt. In den Flash Player-Versionen vor Flash Player 8 wurde die `fscommand()`-Aktion für den Datenaustausch mit der Containeranwendung verwendet. Die ExternalInterface-Klasse ersetzt `fscommand()`.

Hinweis: *Für den Fall, dass Sie die alte `fscommand()`-Funktion verwenden müssen – beispielsweise um die Kompatibilität mit älteren Anwendungen zu wahren oder für die Interaktion mit einer SWF-Containeranwendung von Drittanbietern oder mit dem eigenständigen Flash Player – ist diese weiterhin als Funktion auf Paketebene im flash.system-Paket verfügbar.*

Die ExternalInterface-Klasse ist ein Subsystem, mit dem Sie unkompliziert aus ActionScript und Flash Player mit JavaScript auf einer HTML-Seite kommunizieren können.

Die ExternalInterface-Klasse steht nur in folgenden Fällen zur Verfügung:

- In allen unterstützten Versionen von Internet Explorer für Windows (ab Version 5.0)
- In allen Browsern, die die NPRuntime-Schnittstelle unterstützen, darunter Firefox 1.0 und höher, Mozilla 1.7.5 und höher, Netscape 8.0 und höher und Safari 1.3 und höher
- In einer AIR-Anwendung, wenn die SWF-Datei in eine HTML-Seite eingebettet ist, die vom HTMLLoader-Steuerelement angezeigt wird.

In allen anderen Fällen (z. B. beim Ausführen in einem eigenständigen Player) gibt die `ExternalInterface.available`-Eigenschaft den Wert `false` zurück.

Mit ActionScript können Sie JavaScript-Funktionen in einer HTML-Seite aufrufen. Im Vergleich zu `fscommand()` bietet die externe API die folgende verbesserte Funktionalität:

- Sie können jede gewünschte JavaScript-Funktion verwenden, nicht nur die Funktionen, die mit der `fscommand`-Funktion einsetzbar sind.
- Sie können eine beliebige Anzahl von Argumenten mit frei wählbaren Namen übergeben. Sie sind nicht darauf beschränkt, einen Befehl und einen einzigen Argumentstring zu übergeben. Dadurch bietet die externe API viel mehr Flexibilität als `fscommand()`.
- Sie können unterschiedliche Datentypen übergeben (z. B. Boolean, Number und String); es gibt keine Beschränkung auf Parameter vom Typ String.
- Sie können den Wert eines Aufrufs empfangen. Dieser Wert wird sofort (als Rückgabewert des Aufrufs) an ActionScript zurückgegeben.

Wichtig: *Wenn der Name der Flash Player-Instanz in einer HTML-Seite (das `id`-Attribut des `object`-Tags) einen Bindestrich (-) oder andere Zeichen enthält, die in JavaScript als Operatoren definiert sind (zum Beispiel `+`, `*`, `/`, `\`, `.` usw.), sind ExternalInterface-Aufrufe aus ActionScript beim Anzeigen der Containerwebseite in Internet Explorer unwirksam. Darüber hinaus funktionieren ExternalInterface-Aufrufe aus ActionScript nicht, wenn die die Flash Player-Instanz definierenden HTML-Tags (`object` und `embed`) in einem HTML-`form`-Tag verschachtelt sind.*

Verwenden der ExternalInterface-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Kommunikation zwischen ActionScript und der Containeranwendung kann zwei Formen aufweisen: ActionScript ruft Code auf (z. B. eine JavaScript-Funktion), der im Container definiert ist, oder Code im Container kann eine ActionScript-Funktion aufrufen, die als aufrufbar eingestuft wurde. In beiden Fällen können Informationen an den aufgerufenen Code übergeben und Ergebniswerte an den aufrufenden Code zurückgegeben werden.

Um diesen Datenaustausch zu ermöglichen, enthält die ExternalInterface-Klasse zwei statische Eigenschaften und zwei statische Methoden. Diese Eigenschaften und Methoden werden eingesetzt, um Informationen über die externe Schnittstellenverbindung abzurufen, um Code im Container aus ActionScript auszuführen und um ActionScript-Funktionen für den Aufruf aus dem Container verfügbar zu machen.

Abrufen von Informationen zum externen Container

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `ExternalInterface.available`-Eigenschaft gibt an, ob der aktuelle Flash Player in einem Container ausgeführt wird, der über eine externe Schnittstelle verfügt. Wenn die externe Schnittstelle zur Verfügung steht, lautet der Wert dieser Eigenschaft `true`, ansonsten `false`. Vor dem Einsatz weiterer Funktionen der ExternalInterface-Klasse sollten Sie stets mithilfe der folgenden Vorgehensweise sicherstellen, dass der aktuelle Container die externe Schnittstellenkommunikation unterstützt:

```
if (ExternalInterface.available)
{
    // Perform ExternalInterface method calls here.
}
```

Hinweis: *Mit der `ExternalInterface.available`-Eigenschaft kann festgestellt werden, ob der aktuelle Container die ExternalInterface-Verbindung unterstützt. Es wird nicht angegeben, ob JavaScript im aktuellen Browser aktiviert ist.*

Mit der `ExternalInterface.objectID`-Eigenschaft können Sie die eindeutige ID der Flash Player-Instanz ermitteln (d. h. das `id`-Attribut des `object`-Tags in Internet Explorer oder das `name`-Attribut des `embed`-Tags in Browsern mit der NPRuntime-Schnittstelle). Diese eindeutige ID gibt das aktuelle SWF-Dokument im Browser an und kann für Verweise auf das SWF-Dokument verwendet werden, beispielsweise beim Aufrufen einer JavaScript-Funktion in einer Container-HTML-Seite. Wenn der Flash Player-Container kein Webbrowser ist, hat diese Eigenschaft den Wert `null`.

Abrufen von externem Code aus ActionScript

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `ExternalInterface.call()`-Methode wird Code in der Containeranwendung ausgeführt. Sie erfordert mindestens einen Parameter: einen String mit dem Namen der in der Containeranwendung aufzurufenden Funktion. Alle weiteren an die `ExternalInterface.call()`-Methode übergebenen Parameter werden als Parameter des Funktionsaufrufs an den Container weitergeleitet.

```
// calls the external function "addNumbers"
// passing two parameters, and assigning that function's result
// to the variable "result"
var param1:uint = 3;
var param2:uint = 7;
var result:uint = ExternalInterface.call("addNumbers", param1, param2);
```

Wenn der Container eine HTML-Seite ist, ruft diese Methode die JavaScript-Funktion mit dem angegebenen Namen auf, die in einem `script`-Element der HTML-Seite definiert sein muss. Der Rückgabewert der JavaScript-Funktion wird an ActionScript zurückgegeben.

```
<script language="JavaScript">
    // adds two numbers, and sends the result back to ActionScript
    function addNumbers(num1, num2)
    {
        return (num1 + num2);
    }
</script>
```

Wenn der Container ein anderer ActiveX-Container ist, bewirkt diese Methode, dass das Flash Player-ActiveX-Steuerelement das zugehörige `FlashCall`-Ereignis sendet. Der angegebene Funktionsname und alle Parameter werden in Flash Player in einen XML-String serialisiert. Der Container kann auf diese Informationen über die `request`-Eigenschaft des Ereignisobjekts zugreifen und so ermitteln, in welcher Form der Code ausgeführt werden soll. Für die Rückgabe eines Wertes an ActionScript wird mit dem Containercode die `SetReturnValue()`-Methode des ActiveX-Objekts aufgerufen und der Ergebniswert (in einen XML-String serialisiert) als Parameter dieser Methode übergeben. Weitere Informationen zum für diese Kommunikation verwendeten XML-Format finden Sie unter „XML-Format der externen API“ auf Seite 903.

Unabhängig davon, ob der Container ein Webbrowser oder ein anderer ActiveX-Container ist, wird `null` zurückgegeben, wenn der Aufruf fehlschlägt oder die Containermethode keinen Rückgabewert festlegt. Die `ExternalInterface.call()`-Methode löst eine SecurityError-Ausnahme aus, wenn die Containerumgebung einer Sicherheits-Sandbox angehört, auf die der aufrufende Code keinen Zugriff hat. Sie können dies vermeiden, indem Sie in der Containerumgebung einen entsprechenden Wert für `allowScriptAccess` festlegen. Zum Ändern des Wertes von `allowScriptAccess` in einer HTML-Seite müssen Sie beispielsweise das entsprechende Attribut im `object`-Tag und im `embed`-Tag bearbeiten.

Abrufen von ActionScript-Code aus dem Container

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Container kann nur ActionScript-Code aufrufen, der sich innerhalb einer Funktion befindet. Anderer ActionScript-Code steht für Container nicht zur Verfügung. Um eine ActionScript-Funktion über die Containeranwendung aufzurufen, müssen Sie zwei Dinge tun: Registrieren der Funktion in der ExternalInterface-Klasse und anschließend Aufrufen der Klasse über den Containercode.

Zunächst müssen Sie die ActionScript-Funktion registrieren, um anzugeben, dass sie für die Containeranwendung zur Verfügung steht. Verwenden Sie die `ExternalInterface.addCallback()`-Methode wie folgt:

```
function callMe(name:String):String
{
    return "busy signal";
}
ExternalInterface.addCallback("myFunction", callMe);
```

Für die `addCallback()`-Methode sind zwei Parameter erforderlich. Der erste ist ein String mit dem Funktionsnamen, unter dem die Funktion im Container bekannt ist. Der zweite Parameter ist die eigentliche ActionScript-Funktion, die ausgeführt wird, wenn im Container der definierte Funktionsname aufgerufen wird. Da diese Namen voneinander unabhängig sind, können Sie zur Verwendung für den Container einen Funktionsnamen angeben, der vom eigentlichen Namen der ActionScript-Funktion abweicht. Dies ist besonders hilfreich, wenn der Funktionsname unbekannt ist, beispielsweise wenn eine anonyme Funktion angegeben oder die aufzurufende Funktion zur Laufzeit festgelegt wird.

Nachdem eine ActionScript-Funktion bei der ExternalInterface-Klasse registriert wurde, ist sie aus dem Container aufrufbar. Die Vorgehensweise zum Aufrufen hängt vom Containertyp ab. Beispielsweise werden ActionScript-Funktionen aus JavaScript-Code in einem Webbrowser mithilfe des registrierten Funktionsnamens aufgerufen, als wären sie Methoden des Flash Player-Browserobjekts (d. h. Methoden des JavaScript-Objekts, das das Tag `object` oder `embed` repräsentiert). Das bedeutet, dass Parameter übergeben werden und ein Rückgabewert zurückgegeben wird, wie dies beim Aufrufen einer lokalen Funktion der Fall wäre.

```
<script language="JavaScript">
    // callResult gets the value "busy signal"
    var callResult = flashObject.myFunction("my name");
</script>
...
<object id="flashObject"...>
    ...
    <embed name="flashObject".../>
</object>
```

Im Gegensatz dazu müssen beim Aufrufen einer ActionScript-Funktion in einer SWF-Datei, die in einer Desktopanwendung ausgeführt wird, der registrierte Funktionsname und alle Parameter in einen XML-formatierten String serialisiert werden. Der eigentliche Aufruf erfolgt dann durch Aufrufen der `CallFunction()`-Methode des ActiveX-Steuerelements mit dem XML-String als Parameter. Weitere Informationen zum für diese Kommunikation verwendeten XML-Format finden Sie unter „XML-Format der externen API“ auf Seite 903.

In beiden Fällen wird der Rückgabewert der ActionScript-Funktion an den Containercode zurückgegeben, entweder direkt als Wert (beim Aufruf aus JavaScript-Code in einem Browser) oder serialisiert als XML-formatierter String (beim Aufruf aus einem ActiveX-Container).

XML-Format der externen API

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei der Kommunikation zwischen ActionScript und einer Anwendung, in der ein Shockwave Flash-ActiveX-Steuerelement ausgeführt wird, wird zum Kodieren der Funktionsaufrufe und Rückgabewerte ein spezielles XML-Format verwendet. Es liegen zwei Formen des in der externen API verwendeten XML-Formats vor. Das eine Format wird zum Darstellen von Funktionsaufrufen verwendet. Das andere dient zum Abbilden einzelner Werte. Es wird für Funktionsparameter sowie für Rückgabewerte von Funktionen verwendet. Das XML-Format für Funktionsaufrufe wird für Aufrufe zwischen ActionScript und dem ActiveX-Steuerelement verwendet. Bei einem Funktionsaufruf aus ActionScript wird der XML-String in Flash Player an den Container übergeben. Bei einem Aufruf aus dem Container erwartet Flash Player einen XML-String in diesem Format. Im folgenden XML-Fragment ist ein Beispiel für einen XML-formatierten Funktionsaufruf dargestellt:

```
<invoke name="functionName" returntype="xml">
    <arguments>
        ... (individual argument values)
    </arguments>
</invoke>
```

Der Stammknoten ist der `invoke`-Knoten. Er hat zwei Attribute: `name` gibt den Namen der aufzurufenden Funktion an. `returntype` hat immer den Wert `xml`. Wenn der Funktionsaufruf Parameter enthält, hat der `invoke`-Knoten einen untergeordneten `arguments`-Knoten, dessen untergeordnete Knoten wiederum die Parameterwerte sind. Je nach Wert sind diese individuell formatiert, wie im Folgenden dargestellt ist.

Bei individuellen Werten einschließlich Funktionsnamen und Rückgabewerten wird ein Formatierungsschema verwendet, das zusätzlich zu den eigentlichen Werten auch eine Datentypangabe enthält. In der folgenden Tabelle sind ActionScript-Klassen und das XML-Format aufgeführt, das zum Kodieren der Werte für den jeweiligen Datentyp verwendet wird:

ActionScript-Klasse/-Wert	C#-Klasse/-Wert	Format	Kommentare
null	null	<null/>	
Boolean `true`	bool true	<true/>	
Boolean `false`	bool false	<false/>	
String	string	<string>Stringwert</string>	
Number, int, uint	single, double, int, uint	`<number>27.5</number>` `<number>-12</number>`	

ActionScript-Klasse/-Wert	C#-Klasse/-Wert	Format	Kommentare

Array (Elemente unterschiedlicher Datentypen möglich)	Eine Sammlung, die Elemente unterschiedlicher Datentypen zulässt, z. B. ArrayList oder object[]	`<array>`
		`<property id="0">`
		`<number>27.5</number>`
		`</property>`
		`<property id="1">`
		`<string>Hello there!</string>`
		`</property>`
		`...`
		`</array>`
			Der `property`-Knoten definiert einzelne Elemente. Das `id`-Attribut ist die numerische auf Null basierende Indexposition.

Object	Ein Wörterbuch mit Stringschlüsseln und Objektwerten, z. B. HashTable mit Stringschlüsseln	`<object>`
		`<property id="name">`
		`<string>John Doe</string>`
		`</property>`
		`<property id="age">`
		`<string>33</string>`
		`</property>`
		`...`
		`</object>`
			Der `property`-Knoten definiert einzelne Eigenschaften. Das `id`-Attribut ist der jeweilige Name der Eigenschaft (ein String).

Weitere integrierte oder benutzerdefinierte Klassen		`<null/> or` `<object></object>`	ActionScript kodiert andere Objekte als null oder als ein leeres Objekt. In beiden Fällen gehen alle Eigenschaftswerte verloren.

Hinweis: *Als Beispiel sind in dieser Tabelle zusätzlich zu den ActionScript-Klassen entsprechende C#-Klassen dargestellt. Die externe API kann jedoch eingesetzt werden, um mit beliebigen Programmiersprachen oder Laufzeitumgebungen zu kommunizieren, die ActiveX-Steuerelemente unterstützen. Sie ist nicht auf C#-Anwendungen beschränkt.*

Wenn Sie Anwendungen mithilfe der externen API und einem ActiveX-Container erstellen, ist es möglicherweise hilfreich, eine Proxy-Klasse zu programmieren, mit der die jeweiligen Funktionsaufrufe in das serialisierte XML-Format konvertiert werden. Ein Beispiel einer in C# geschriebeben Proxy-Klasse finden Sie unter Details der ExternalInterfaceProxy-Klasse.

Beispiel für externe API: Kommunikation zwischen ActionScript und JavaScript in einem Webbrowser

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Diese Beispielanwendung veranschaulicht geeignete Techniken für den Datenaustausch zwischen ActionScript und JavaScript in einem Webbrowser im Kontext einer Instant Messaging-Anwendung, die es einer Person ermöglicht, mit sich selbst zu chatten (daher der Name der Anwendung: Introvert IM). Nachrichten werden mithilfe der externen API zwischen einem in die Webseite integrierten HTML-Formular und einer SWF-Benutzeroberfläche ausgetauscht. In diesem Beispiel werden folgende Techniken demonstriert:

- Ordnungsgemäßes Initiieren des Datenaustauschs durch Überprüfen, ob der Browser bereit ist, bevor ein Datenaustausch gestartet wird
- Überprüfen, ob der Container die externe API unterstützt
- Aufrufen von JavaScript-Funktionen aus ActionScript, Übergeben von Parametern und Empfangen von Rückgabewerten

- Verfügbarmachen von ActionScript-Methoden für Aufrufe aus JavaScript und Ausführen dieser Aufrufe

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „Introvert IM“ befinden sich im Ordner „Samples/IntrovertIM_HTML“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
IntrovertIMApp.fla oder IntrovertIMApp.mxml	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/introvertIM/IMManager.as	Die Klasse, mit der der Datenaustausch zwischen ActionScript und dem Container aufgebaut und verwaltet wird.
com/example/programmingas3/introvertIM/IMMessageEvent.as	Ein benutzerdefinierter Ereignistyp, der von der IMManager-Klasse ausgelöst wird, wenn eine vom Container stammende Nachricht empfangen wurde.
com/example/programmingas3/introvertIM/IMStatus.as	Eine Aufzählung, deren Werte die verschiedenen Statuswerte für die „Verfügbarkeit“ angeben, die in der Anwendung ausgewählt werden kann.
html-flash/IntrovertIMApp.html oder html-template/index.template.html	Für Flash die HTML-Seite der Anwendung (html-flash/IntrovertIMApp.html) oder für Adobe Flex die Vorlage, die zum Erstellen der Container-HTML-Seite der Anwendung verwendet wird (html-template/index.template.html). Diese Datei enthält alle JavaScript-Funktionen der Containerkomponente der Anwendung.

Vorbereiten der Kommunikation mit dem ActionScript-Browser

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Einer der häufigsten Einsatzzwecke für die externe API ist der Datenaustausch zwischen ActionScript-Anwendungen und einem Webbrowser. Mithilfe der externen API kann mit ActionScript-Methoden JavaScript-Code aufgerufen werden und umgekehrt. Aufgrund der Komplexität von Browsern und der Art, wie diese Seiten intern darstellen, gibt es keine Möglichkeit sicherzustellen, dass ein SWF-Dokument die zugehörigen Callback-Funktionen registriert hat, bevor die erste JavaScript-Anweisung in der HTML-Seite ausgeführt wird. Aus diesem Grund sollte das SWF-Dokument stets zuerst die HTML-Seite aufrufen und darüber benachrichtigen, dass es verbindungsbereit ist, bevor aus JavaScript Funktionen des SWF-Dokuments aufgerufen werden.

In der Anwendung „Introvert IM“ wird beispielsweise durch eine Reihe von Schritten in der IMManager-Klasse ermittelt, ob der Browser für den Datenaustausch bereit ist, und die SWF-Datei dann darauf vorbereitet. Der erste Schritt – Ermitteln, ob der Browser für den Datenaustausch bereit ist – erfolgt im IMManager-Konstruktor:

```
public function IMManager(initialStatus:IMStatus)
{
    _status = initialStatus;

    // Check if the container is able to use the external API.
    if (ExternalInterface.available)
    {
        try
        {
            // This calls the isContainerReady() method, which in turn calls
            // the container to see if Flash Player has loaded and the container
            // is ready to receive calls from the SWF.
            var containerReady:Boolean = isContainerReady();
            if (containerReady)
            {
                // If the container is ready, register the SWF's functions.
                setupCallbacks();
            }
            else
            {
                // If the container is not ready, set up a Timer to call the
                // container at 100ms intervals. Once the container responds that
                // it's ready, the timer will be stopped.
                var readyTimer:Timer = new Timer(100);
                readyTimer.addEventListener(TimerEvent.TIMER, timerHandler);
                readyTimer.start();
            }
        }
        ...
    }
    else
    {
        trace("External interface is not available for this container.");
    }
}
```

Zunächst wird mithilfe der `ExternalInterface.available`-Eigenschaft überprüft, ob die externe API im aktuellen Container überhaupt verfügbar ist. Wenn dies der Fall ist, beginnt der Aufbau der Kommunikation. Da beim Kommunikationsaufbau mit externen Anwendungen Sicherheitsausnahmen und andere Fehler auftreten können, ist der Code in einen `try`-Block eingeschlossen. (Die entsprechenden `catch`-Blöcke sind aus Platzgründen nicht aufgeführt.)

Anschließend wird die im Folgenden aufgeführte `isContainerReady()`-Methode aufgerufen:

```
private function isContainerReady():Boolean
{
    var result:Boolean = ExternalInterface.call("isReady");
    return result;
}
```

Die `isContainerReady()`-Methode verwendet wiederum wie folgt die `ExternalInterface.call()`-Methode, um die JavaScript-Funktion `isReady()` aufzurufen:

```
<script language="JavaScript">
<!--
// ------- Private vars -------
var jsReady = false;
...
// ------- functions called by ActionScript -------
// called to check if the page has initialized and JavaScript is available
function isReady()
{
    return jsReady;
}
...
// called by the onload event of the <body> tag
function pageInit()
{
    // Record that JavaScript is ready to go.
    jsReady = true;
}
...
//-->
</script>
```

Die `isReady()`-Funktion gibt den Wert der Variablen `jsReady` zurück. Diese Variable erhält zu Anfang den Wert `false`. Nachdem das `onload`-Ereignis der Webseite ausgelöst wurde, ist der Wert der Variablen `true`. Mit anderen Worten: Wenn ActionScript die `isReady()`-Funktion aufruft, bevor die Seite geladen ist, gibt JavaScript beim Aufruf von `ExternalInterface.call("isReady")` den Wert `false` zurück. Daraufhin gibt auch die ActionScript-Methode `isContainerReady()` den Wert `false` zurück. Nachdem die Seite geladen wurde, gibt die JavaScript-Funktion `isReady()` den Wert `true` zurück, daraufhin gibt auch die ActionScript-Methode`isContainerReady` den Wert `true` zurück.

Je nach Bereitschaft des Containers werden im IMManager-Konstruktor zwei verschiedene Aktionen ausgeführt. Wenn `isContainerReady()` den Wert `true` zurückgibt, wird die `setupCallbacks()`-Methode aufgerufen, die den Aufbau der Kommunikation mit JavaScript abschließt. Wenn `isContainerReady()` jedoch `false` zurückgibt, wird der Vorgang vorerst ausgesetzt. Es wird wie folgt ein Timer-Objekt erstellt, das alle 100Millisekunden die `timerHandler()`-Methode aufrufen soll:

```
private function timerHandler(event:TimerEvent):void
{
    // Check if the container is now ready.
    var isReady:Boolean = isContainerReady();
    if (isReady)
    {
        // If the container has become ready, we don't need to check anymore,
        // so stop the timer.
        Timer(event.target).stop();
        // Set up the ActionScript methods that will be available to be
        // called by the container.
        setupCallbacks();
    }
}
```

Bei jedem Aufruf der `timerHandler()`-Methode wird erneut das Ergebnis der `isContainerReady()`-Methode überprüft. Wenn der Container initialisiert ist, gibt diese Methode Folgendes zurück: `true`. Daraufhin wird der Timer angehalten und die `setupCallbacks()`-Methode aufgerufen, um den Aufbau der Kommunikation mit dem Browser abzuschließen.

Bereitstellen von ActionScript-Methoden für JavaScript

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wie im vorangegangenen Beispiel gezeigt wurde, wird die `setupCallbacks()`-Methode aufgerufen, nachdem ermittelt wurde, dass der Browser bereit ist. Mit dieser Methode wird ActionScript wie folgt darauf vorbereitet, Aufrufe aus JavaScript zu empfangen:

```
private function setupCallbacks():void
{
    // Register the SWF client functions with the container
    ExternalInterface.addCallback("newMessage", newMessage);
    ExternalInterface.addCallback("getStatus", getStatus);
    // Notify the container that the SWF is ready to be called.
    ExternalInterface.call("setSWFIsReady");
}
```

Die `setCallBacks()`-Methode schließt die Vorbereitung der Kommunikation mit dem Container ab, indem `ExternalInterface.addCallback()` aufgerufen wird, um die beiden Methoden zu registrieren, die für den Aufruf aus JavaScript verfügbar sein sollen. Im Codebeispiel stimmt der erste Parameter – der Name, unter dem die Methode in JavaScript bekannt ist (`„newMessage"` und `„getStatus"`) – mit dem Namen der Methode in ActionScript überein. (In diesem Fall brachte es keine Vorteile, unterschiedliche Namen zu verwenden, deshalb wurde aus Gründen der Einfachheit derselbe Name verwendet.) Schließlich wird die `ExternalInterface.call()`-Methode verwendet, um die JavaScript-Funktion `setSWFIsReady()` aufzurufen, die den Container darüber benachrichtigt, dass die ActionScript-Funktionen nun registriert sind.

Datenübertragung von ActionScript zum Browser

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In der Anwendung „Introvert IM“ werden viele Beispiele für das Aufrufen von JavaScript-Funktionen in der Containerseite vorgestellt. Im einfachsten Fall (ein Beispiel aus der `setupCallbacks()`-Methode) wird die JavaScript-Funktion `setSWFIsReady()` aufgerufen, ohne dass Parameter übergeben oder ein Rückgabewert zurückgegeben wird:

```
ExternalInterface.call("setSWFIsReady");
```

In einem anderen Beispiel aus der `isContainerReady()`-Methode ruft ActionScript die `isReady()`-Funktion auf und erhält als Rückgabewert einen booleschen Wert:

```
var result:Boolean = ExternalInterface.call("isReady");
```

Sie können mithilfe der externen API auch Parameter an JavaScript-Funktionen übergeben. Dies erfolgt beispielsweise mit der `sendMessage()`-Methode der IMManager-Klasse, die aufgerufen wird, wenn der Benutzer eine neue Nachricht an den „Gesprächspartner“ sendet:

```
public function sendMessage(message:String):void
{
    ExternalInterface.call("newMessage", message);
}
```

Wieder wird `ExternalInterface.call()` verwendet, um die entsprechende JavaScript-Funktion aufzurufen, mit der der Browser über die neue Nachricht informiert wird. Zusätzlich wird die Nachricht selbst als weiterer Parameter an `ExternalInterface.call()` übergeben und dann als Parameter an die JavaScript-Funktion `newMessage()` weitergeleitet.

Aufrufen von ActionScript-Code aus JavaScript

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine wirkliche Kommunikation findet erst statt, wenn der Datenaustausch in beide Richtungen erfolgt, und die Anwendung „Introvert IM“ stellt dabei keine Ausnahme dar. Nicht nur der Flash Player-IM-Client ruft JavaScript-Funktionen auf, um Nachrichten zu senden, sondern auch das HTML-Formular verwendet JavaScript, um Nachrichten an die SWF-Datei zu senden und Informationen abzufragen. Wenn beispielsweise die SWF-Datei den Container benachrichtigt, dass die Kontaktaufnahme abgeschlossen und Kommunikationsbereitschaft hergestellt ist, ruft der Browser zunächst die `getStatus()`-Methode der IMManager-Klasse auf, um vom SWF-IM-Client den Anfangsstatus der Benutzerverfügbarkeit abzurufen. Dies erfolgt mithilfe der `updateStatus()`-Funktion in der Webseite:

```
<script language="JavaScript">
...
function updateStatus()
{
    if (swfReady)
    {
        var currentStatus = getSWF("IntrovertIMApp").getStatus();
        document.forms["imForm"].status.value = currentStatus;
    }
}
...
</script>
```

Es wird der Wert der Variablen `swfReady` überprüft, in der gespeichert ist, ob die SWF-Datei den Browser darüber benachrichtigt hat, dass die Methoden nun bei der ExternalInterface-Klasse registriert sind. Wenn die SWF-Datei kommunikationsbereit ist, wird mit der nächsten Zeile (`var currentStatus =` ...) die `getStatus()`-Methode der IMManager-Klasse aufgerufen. In dieser Codezeile werden drei Vorgänge durchgeführt:

- Die JavaScript-Funktion `getSWF()` wird aufgerufen, die einen Verweis auf das JavaScript-Objekt zurückgibt, mit dem die SWF-Datei angegeben wird. Mit dem an `getSWF()` übergebenen Parameter wird festgelegt, welches Browserobjekt zurückgegeben wird, wenn pro HTML-Seite mehrere SWF-Dateien vorhanden sind. Der Wert dieses Parameters muss mit dem `id`-Attribut des `object`-Tags und dem `name`-Attribut des `embed`-Tags übereinstimmen, die zum Einfügen der SWF-Datei verwendet wurden.
- Mithilfe des Verweises auf die SWF-Datei wird die `getStatus()`-Methode aufgerufen, als wäre sie eine Methode des SWF-Objekts. In diesem Fall wird der Funktionsname `getStatus` verwendet, da die ActionScript-Funktion mithilfe von `ExternalInterface.addCallback()` unter diesem Namen registriert wurde.
- Die ActionScript-Methode `getStatus()` gibt einen Wert zurück. Dieser Wert wird der Variablen `currentStatus` zugewiesen, die dann als Inhalt des Textfelds `status` (über die `value`-Eigenschaft) zugewiesen wird.

Hinweis: *Wenn Sie den Code mitverfolgt haben, ist Ihnen wahrscheinlich aufgefallen, dass im Quellcode für die Funktion `updateStatus()` die Codezeile, die die Funktion `getSWF()` aufruft, folgendermaßen geschrieben ist: var currentStatus = getSWF("${application}").getStatus(); Der Text `${application}` ist ein Platzhalter in der HTML-Seitenvorlage. Wenn Adobe Flash Builder die tatsächliche HTML-Seite für die Anwendung generiert, wird dieser Platzhalter durch den Text ersetzt, der für das `object`-Tag als `id`-Attribut und für das `embed`-Tag als `name`-Attribut verwendet wird (in diesem Beispiel `IntrovertIMApp`). Dies ist der Wert, der von der `getSWF()`-Funktion erwartet wird.*

Die JavaScript-Funktion `sendMessage()` veranschaulicht das Übergeben von Parametern an eine ActionScript-Funktion. (Die `sendMessage()`-Funktion wird aufgerufen, wenn der Benutzer auf der HTML-Seite auf die Schaltfläche „Send“ klickt.)

```
<script language="JavaScript">
...
function sendMessage(message)
{
    if (swfReady)
    {
        ...
        getSWF("IntrovertIMApp").newMessage(message);
    }
}
...
</script>
```

Die ActionScript-Methode `newMessage()` erwartet einen Parameter, deshalb wird die JavaScript-Variable `message` an ActionScript übergeben, indem sie im JavaScript-Code beim Aufruf der `newMessage()`-Methode als Parameter verwendet wird.

Erkennen des Browsertyps

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Aufgrund der browserspezifischen Unterschiede beim Zugriff auf Inhalte ist es wichtig, immer den vom Benutzer verwendeten Browser zu ermitteln und auf den Film der jeweiligen Syntax entsprechend zuzugreifen. Dies erfolgt mithilfe von JavaScript und des Window- oder Document-Objekts, wie in der JavaScript-Funktion `getSWF()` in diesem Beispiel dargestellt ist:

```
<script language="JavaScript">
...
function getSWF(movieName)
{
    if (navigator.appName.indexOf("Microsoft") != -1)
    {
        return window[movieName];
    }
    else
    {
        return document[movieName];
    }
}
...
</script>
```

Wenn der Browsertyp des Benutzers im Skript nicht erkannt wird, kann es bei der Wiedergabe von SWF-Dateien in einem HTML-Container zu unerwartetem Verhalten kommen.

Kapitel 48: Validierung von XML-Signaturen in AIR

Adobe AIR 1.5 und höher

Die Klassen der XMLSignatureValidator-API von Adobe® AIR® dienen zur Validierung von digitalen Signaturen gemäß einem Teilsatz der W3C-Empfehlung für die Syntax und Verarbeitung von XML-Signaturen (http://http://www.w3.org/TR/xmldsig-core/). Mit XML-Signaturen kann die Integrität und die Identität des Unterzeichners von Daten oder Informationen überprüft werden.

Mit XML-Signaturen können Meldungen oder Ressourcen, die von Ihrer Anwendung heruntergeladen wurden, validiert werden. Wenn Ihre Anwendung zum Beispiel Dienste auf Abonnementsbasis bereitstellt, können Sie die Abo-Bedingungen in ein signiertes XML-Dokument einbetten. Wenn eine Person versucht, das Abonnementsdokument zu ändern, schlägt die Validierung fehl.

Sie können mit einer XML-Signatur auch heruntergeladene Ressourcen, die von Ihrer Anwendung verwendet werden, validieren, indem Sie ein signiertes Manifest mit Digests dieser Ressourcen einschließen. Ihre Anwendung kann verifizieren, dass die Ressourcen nicht geändert wurden, indem der Digest in der signierten Datei mit einem aus den geladenen Byte berechneten Digest verglichen wird. Dies ist besonders hilfreich, wenn es sich bei der heruntergeladenen Ressource um eine SWF-Datei oder um andere interaktive Inhalte handelt, die in der Anwendungs-Sandbox ausgeführt werden sollen.

Grundlagen zur Validierung von XML-Signaturen

Adobe AIR 1.5 und höher

Eine Kurzbeschreibung und Codebeispiele zur Validierung von XML-Signaturen finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Erstellen und Validieren von XML-Signaturen (Flex)
- Erstellen und Validieren von XML-Signaturen (Flash)

Adobe® AIR® stellt die XMLSignatureValidator-Klasse und die IURIDereferencer-Schnittstelle zum Validieren von XML-Signaturen zur Verfügung. Die von der XMLSignatureValidator-Klasse akzeptierte XML-Syntax ist ein Teilsatz der W3C-Empfehlung für die Syntax und Verarbeitung von XML-Signaturen. (Da nur ein Teilsatz der Empfehlung unterstützt wird, können nicht alle zulässigen Signaturen validiert werden.) AIR stellt keine API zum Erstellen von XML-Signaturen bereit.

Validierungsklassen für XML-Signaturen

Adobe AIR 1.5 und höher

Die Validierungs-API für XML-Signaturen enthält die folgenden Klassen:

Paket	Klassen
flash.security	• XMLSignatureValidator • IURIDereferencer (Schnittstelle) XMLSignatureValidator-Stringkonstanten werden in den folgenden Klassen definiert: • ReferencesValidationSetting • RevocationCheckSettings • SignatureStatus • SignerTrustSettings
flash.events	• Event • ErrorEvent

Verwendung der Validierungsklassen für XML-Signaturen

Adobe AIR 1.5 und höher

Sie müssen Folgendes ausführen, wenn Sie eine XML-Signatur mit der XMLSignatureValidator-Klasse validieren möchten:

- Erstellen Sie ein XMLSignatureValidator-Objekt.
- Stellen Sie eine Implementierung der IURIDereferencer-Schnittstelle bereit. Das XMLSignatureValidator-Objekt ruft die IURIDereferencer-Methode `dereference()` auf, wobei der URI für jeden Verweis in einer Signatur übergeben wird. Die `dereference()`-Methode muss den URI auflösen und die referenzierten Daten zurückgegeben (diese können sich in demselben Dokument wie die Signatur oder in einer externen Ressource befinden).
- Legen Sie die Einstellungen für Zertifikatsvertrauenswürdigkeit, Sperrungsüberprüfung und Referenzvalidierung des XMLSignatureValidator-Objekt auf für Ihre Anwendung geeignete Werte fest.
- Fügen Sie Ereignis-Listener für die `complete`- und `error`-Ereignisse hinzu.
- Rufen Sie die `verify()`-Methode auf, wobei Sie die zu verifizierende Signatur übergeben.
- Verarbeiten Sie die `complete`- und `error`-Ereignisse und interpretieren Sie die Ergebnisse.

Im folgenden Beispiel wird eine `validate()`-Funktion implementiert, die die Gültigkeit einer XML-Signatur überprüft. Die XMLSignatureValidator-Eigenschaften werden so eingestellt, dass das signierende Zertifikat sich im Vertrauensspeicher des Systems befinden muss, oder es muss eine Verknüpfung mit einem Zertifikat im Vertrauensspeicher bestehen. Im Beispiel wird davon ausgegangen, dass eine geeignete IURIDereferencer-Klasse mit dem Namen *XMLDereferencer* vorhanden ist.

```
private function validate( xmlSignature:XML ):void
{
    var verifier:XMLSignatureValidator = new XMLSignatureValidator();
    verifier.addEventListener(Event.COMPLETE, verificationComplete);
    verifier.addEventListener(ErrorEvent.ERROR, verificationError);
    try
    {
        verifier.uriDereferencer = new XMLDereferencer();

        verifier.referencesValidationSetting =
            ReferencesValidationSetting.VALID_IDENTITY;
        verifier.revocationCheckSetting = RevocationCheckSettings.BEST_EFFORT;
        verifier.useSystemTrustStore = true;

        //Verify the signature
        verifier.verify( xmlSignature );
    }
    catch (e:Error)
        {
            trace("Verification error.\n" + e);
        }
}

//Trace verification results
private function verificationComplete(event:Event):void

    var signature:XMLSignatureValidator = event.target as XMLSignatureValidator;
    trace("Signature status: " + signature.validityStatus + "\n");
    trace("  Digest status: " + signature.digestStatus + "\n");
    trace("  Identity status: " + signature.identityStatus + "\n");
    trace("  Reference status: " + signature.referencesStatus + "\n");
}

private function verificationError(event:ErrorEvent):void
{
    trace("Verification error.\n" + event.text);
}
```

Ablauf der XML-Signaturvalidierung

Adobe AIR 1.5 und höher

Wenn Sie die XMLSignatureValidator-Methode `verify()` aufrufen, führt AIR die folgenden Schritte aus:

- Die Laufzeitumgebung überprüft die kryptografische Integrität der Signatur mithilfe des öffentlichen Schlüssels des signierenden Zertifikats.
- Die Laufzeitumgebung stellt die kryptografische Integrität, Identität und Vertrauenswürdigkeit des Zertifikats anhand der aktuellen Einstellungen des XMLSignatureValidator-Objekts her.

  Das in das signierende Zertifikat gesetzte Vertrauen spielt eine wichtige Rolle für die Integrität des Validierungsprozesses. Die Signaturvalidierung wird mit einem gut definierten kryptografischen Prozess durchgeführt, die Vertrauenswürdigkeit des signierenden Zertifikats kann jedoch nicht algorithmisch beurteilt werden.

  Im Allgemeinen haben Sie drei Möglichkeiten, über die Vertrauenswürdigkeit eines Zertifikats zu entscheiden:

  - Sie verlassen sich auf die Zertifizierungsstellen und den Vertrauensspeicher des Betriebssystems.

- Sie erhalten direkt von Signaturgeber eine Kopie des Zertifikats, ein anderes Zertifikat, das als eine Art Bürge für das Zertifikat dient, oder ausreichende Informationen, um das Zertifikat zuverlässig zu identifizieren, zum Beispiel den öffentlichen Schlüssel.
- Sie fragen den Endbenutzer Ihrer Anwendung, ob er das Zertifikat für vertrauenswürdig hält. Eine solche Abfrage ist bei selbst signierten Zertifikaten ungültig, da die identifizierenden Informationen im Zertifikat an sich unzuverlässig sind.

- Die Laufzeitumgebung überprüft die kryptografische Integrität der signierten Daten.

  Die signierten Daten werden mithilfe Ihrer IURIDereferencer-Implementierung überprüft. Für jeden Verweis im Signaturdokument wird die `dereference()`-Methode des IURIDereferencer-Objekts aufgerufen. Mit den von der `dereference()`-Methode zurückgegebenen Daten wird das Verweisdigest berechnet. Dieser Digestwert wird mit dem im Signaturdokument aufgezeichneten Digest verglichen. Wenn die Digests übereinstimmen, wurden die Daten seit dem Signieren nicht geändert.

  Wenn Sie sich auf die Ergebnisse der Validierung einer XML-Signatur verlassen, ist eine wichtige Überlegung, dass nur das, was signiert ist, sicher ist. Stellen Sie sich zum Beispiel ein signiertes Manifest vor, das die Dateien in einem Paket auflistet. Wenn der XMLSignatureValidator die Signatur verifiziert, wird nur überprüft, ob das Manifest selbst unverändert ist. Die Daten in den Dateien sind nicht signiert, weshalb die Signatur immer noch als valide erkannt wird, wenn die Dateien, auf die im Manifest verwiesen wird, geändert oder gelöscht wurden.

  Hinweis: *Um die Dateien in einem derartigen Manifest zu überprüfen, können Sie den Digest der Dateidaten berechnen (mit demselben Hashing-Algorithmus, der im Manifest verwendet wird) und das Ergebnis mit dem im signierten Manifest gespeicherten Digest vergleichen. In bestimmten Fällen sollten Sie auch prüfen, ob zusätzliche Dateien vorhanden sind.*

Interpretation der Validierungsergebnisse

Adobe AIR 1.5 und höher

Die Validierungsergebnisse werden von den Statuseigenschaften des XMLSignatureValidator-Objekts gemeldet. Diese Eigenschaften können gelesen werden, nachdem das Validatorobjekt das *complete*-Ereignis ausgelöst hat. Die vier Statuseigenschaften sind: `validityStatus`, `digestStatus`, `identityStatus` und `referencesStatus`.

Die validityStatus-Eigenschaft

Adobe AIR 1.5 und höher

Die `validityStatus`-Eigenschaft meldet die allgemeine Gültigkeit der Signatur. Die `validityStatus`-Eigenschaft ist von den anderen drei Statuseigenschaften abhängig und kann einen der folgenden Werte annehmen:

- `valid` – Wenn `digestStatus`, `identityStatus` und `referencesStatus` alle den Wert `valid` aufweisen.
- `invalid` – Wenn mindestens eine der einzelnen Statuseigenschaften den Wert `invalid` aufweist.
- `unknown` – Wenn mindestens eine der einzelnen Statuseigenschaften den Wert `unknown` aufweist und keine einzelne Statuseigenschaft den Wert `invalid` aufweist.

Die digestStatus-Eigenschaft

Adobe AIR 1.5 und höher

Die `digestStatus`-Eigenschaft meldet die Ergebnisse der kryptografischen Überprüfung des Meldungsdigests. Die `digestStatus`-Eigenschaft kann einen der folgenden Werte annehmen:

- `valid` – Wenn das Signaturdokument selbst seit dem Signieren nicht geändert wurde.

- `invalid` – Wenn das Signaturdokument geändert wurde oder fehlerhaft strukturiert ist.
- `unknown` – Wenn die `verify()`-Methode nicht ohne Fehler abgeschlossen wurde.

Die identityStatus-Eigenschaft

Adobe AIR 1.5 und höher

Die `identityStatus`-Eigenschaft meldet den Status des signierenden Zertifikats. Der Wert dieser Eigenschaft ist von verschiedenen Faktoren abhängig, darunter:

- die kryptografische Integrität des Zertifikats
- ob das Zertifikat abgelaufen oder gesperrt ist
- ob das Zertifikat auf dem aktuellen Computer als vertrauenswürdig eingestuft wird
- der Status des XMLSignatureValidator-Objekts (zum Beispiel ob weitere Zertifikate hinzugefügt wurden, um die Vertrauenskette aufzubauen, ob diese Zertifikate vertrauenswürdig sind, und die Werte der `useSystemTrustStore`- und `revocationCheckSettings`-Eigenschaften)

Die `identityStatus`-Eigenschaft kann die folgenden Werte annehmen:

- `valid` – Um als gültig zu gelten, muss das signierende Zertifikat die folgenden Bedingungen erfüllen:
  - Das signierende Zertifikat darf nicht verändert worden sein.
  - Das signierende Zertifikat darf nicht abgelaufen oder gesperrt sein, es sei denn, es befindet sich ein gültiger Zeitstempel in der Signatur. Wenn die Signatur über einen Zeitstempel verfügt, gilt das Zertifikat als gültig, falls es zum Zeitpunkt des Signierens gültig war. (Das Zertifikat, das vom Zeitstempeldienst zum Signieren des Zeitstempels verwendet wird, muss mit einem vertrauenswürdigen Stammzertifikat auf dem Computer des Benutzers verkettet sein.)
  - Das signierende Zertifikat ist vertrauenswürdig. Ein Zertifikat wird als vertrauenswürdig eingestuft, wenn sich das Zertifikat im Vertrauensspeicher des Systems befindet oder mit einem anderen Zertifikat im Vertrauensspeicher des Systems verknüpft ist und Sie die `useSystemTrustStore`-Eigenschaft auf „true“ einstellen. Sie können ein Zertifikat auch als vertrauenswürdig einstufen, indem Sie die `addCertificate()`-Methode des XMLSignatureValidator-Objekts verwenden.
  - Das Zertifikat ist tatsächlich das signierende Zertifikat.
- `invalid` – Das Zertifikat ist abgelaufen oder gesperrt (und es ist kein Zeitstempel vorhanden, der die Gültigkeit zum Zeitpunkt des Signierens bestätigt) oder das Zertifikat wurde geändert.
- `unknown` – Wenn das Zertifikat zwar nicht ungültig, aber auch nicht vertrauenswürdig ist. Selbst signierte Zertifikate werden zum Beispiel als `unknown` gemeldet (sofern sie nicht ausdrücklich als vertrauenswürdig eingestuft sind). Die `identityStatus`-Eigenschaft bekommt auch dann den Wert `unknown`, wenn die `verify()`-Methode nicht ohne Fehler abgeschlossen wurde oder wenn die Identität nicht überprüft wurde, weil der Signaturdigest ungültig ist.

Die referencesStatus-Eigenschaft

Adobe AIR 1.5 und höher

Die `referencesStatus`-Eigenschaft meldet die kryptografische Integrität der Verweise im SignedData-Element der Signatur.

- `valid` – Wenn der berechnete Digest aller Verweise in der Signatur mit dem entsprechenden in der XML-Signatur aufgezeichneten Digest übereinstimmt. Der Status `valid` gibt an, dass die signierten Daten nicht geändert wurden.

- `invalid` – Wenn ein berechneter Digest nicht mit dem entsprechenden Digest in der Signatur übereinstimmt.
- `unknown` – Wenn die Verweisdigests nicht überprüft wurden. Die Verweise werden nicht überprüft, wenn der allgemeine Signaturdigest den Wert `invalid` hat oder das signierende Zertifikat ungültig ist. Falls `identityStatus` den Wert `unknown` aufweist, werden die Verweise nur überprüft, wenn `referencesValidationSetting` den Wert `validOrUnknown` aufweist.

Über XML-Signaturen

Adobe AIR 1.5 und höher

Eine XML-Signatur ist eine digitale Signatur in der XML-Syntax. Mithilfe der Daten in einer XML-Signatur kann überprüft werden, ob die signierten Informationen seit dem Signieren geändert wurden. Wenn ein signierendes Zertifikat von einer vertrauenswürdigen Zertifizierungsstelle ausgegeben wurde, kann über die Public-Key-Infrastruktur (PKI) die Identität des Signaturgebenden verifiziert werden.

Eine XML-Signatur kann auf beliebige digitale Daten (im Binär- oder XML-Format) angewendet werden. Typische Verwendungszwecke für XML-Signaturen sind zum Beispiel:

- Überprüfen, ob externe oder heruntergeladene Daten geändert wurden
- Bestätigen, dass Meldungen aus einer bekannten Quelle stammen
- Validieren von Anwendungslizenzen oder Abonnementberechtigungen

Unterstützte XML-Signatursyntax

Adobe AIR 1.5 und höher

AIR unterstützt die folgenden Elemente aus der W3C-Empfehlung für die Syntax und Verarbeitung von XML-Signaturen:

- Alle Kernelemente der Signatursyntax (Abschnitt 4 der W3C-Empfehlung); nur das KeyInfo-Element wird nicht vollständig unterstützt
- Das KeyInfo-Element darf nur ein X509Data-Element enthalten
- Ein X509Data-Element darf nur ein X509Certificate-Element enthalten
- Die SHA256-Digestmethode
- Der RSA-SHA1-Signaturalgorithmus (PKCS1)
- Die „Canonical XML without comments“-Kanonisierungsmethode und Transformation
- Die umhüllte Signaturtransformation
- Zeitstempel

Das folgende Beispiel veranschaulicht eine typische XML-Signatur (die meisten kryptografischen Daten wurden der Einfachheit halber entfernt):

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <Reference URI="URI_to_signed_data">
            <Transforms>
                <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/></Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>uoo...vY=</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>Ked...w==</SignatureValue>
    <KeyInfo>
        <X509Data>
            <X509Certificate>i7d...w==</X509Certificate>
        </X509Data>
    </KeyInfo>
</Signature>
```

Die Hauptelemente einer Signatur sind:

- SignedInfo – Enthält Verweise auf die signierten Daten und die berechneten Digestwerte zum Zeitpunkt des Signierens. Die signierten Daten können sich in demselben Dokument wie die XML-Signatur oder in einem externen Dokument befinden.
- SignatureValue – Enthält ein Digest des SignedInfo-Elements, das mit dem privaten Schlüssel des Signaturgebers verschlüsselt wurde.
- KeyInfo – Enthält das signierende Zertifikat sowie ggf. zusätzliche Zertifikate, um die Vertrauenskette aufzubauen. Beachten Sie, dass das KeyInfo-Element technisch zwar nicht erforderlich ist, AIR die Signatur bei seinem Fehlen jedoch nicht validieren kann.

Es gibt drei allgemeine XML-Signaturtypen:

- Umhüllt – Die Signatur wird in die XML-Daten, die signiert werden, eingefügt.
- Umhüllend – Die signierten XML-Daten sind in einem Object-Element innerhalb des Signature-Elements enthalten.
- Getrennt – Die signierten Daten befinden sich außerhalb der XML-Signatur. Die signierten Daten können in einer externen Datei enthalten sein. Alternativ dazu können sie sich auch in demselben XML-Dokument wie die Signatur, aber nicht als über- oder untergeordnetes Element des Signature-Elements befindet.

XML-Signaturen verweisen mit URIs auf die signierten Daten. Die signierende und die validierende Anwendung müssen dieselben Konventionen zur Auflösung dieser URIs verwenden. Bei Verwendung der XMLSignatureValidator-Klasse müssen Sie eine Implementierung der IURIDereferencer-Schnittstelle bereitstellen. Diese Implementierung ist für die Auflösung der URIs und für die Rückgabe der signierten Daten als ByteArray-Objekt zuständig. Das zurückgegebene ByteArray-Objekt wird mit demselben Algorithmus verarbeitet, der den Digest in der Signatur produziert hat.

Zertifikate und Vertrauen

Adobe AIR 1.5 und höher

Ein Zertifikat besteht aus einem öffentlichen Schlüssel (Public Key) und ggf. einem oder mehreren Zertifikaten, die zu der ausgebenden Zertifizierungsstelle gehören.

Es gibt zwei Möglichkeiten, in einem Zertifikat Vertrauen herzustellen. Sie können eine Kopie des Zertifikats direkt vom Signaturgeber erhalten, zum Beispiel auf einem physischen Datenträger oder über eine sichere digitale Übertragung wie eine SSL-Transaktion. Sie können sich auch auf eine Zertifizierungsstelle verlassen, um festzustellen, ob das signierende Zertifikat vertrauenswürdig ist.

Damit Sie sich auf eine Zertifizierungsstelle verlassen können, muss das signierende Zertifikat von einer Zertifizierungsstelle ausgegeben worden sein, die auf dem Computer, auf dem die Signatur validiert wird, als vertrauenswürdig eingestuft wird. Die meisten Hersteller von Betriebssystemen platzieren die Stammzertifikate einer Anzahl von Zertifizierungsstellen im Vertrauensspeicher des Betriebssystems. Benutzer können diesem Speicher Zertifikate hinzufügen und sie daraus entfernen.

Auch wenn ein Zertifikat von einer vertrauenswürdigen Zertifizierungsstelle ausgegeben wurde, müssen Sie immer noch entscheiden, ob das Zertifikat jemandem gehört, dem Sie vertrauen. In vielen Fällen wird diese Entscheidung dem Endbenutzer überlassen. Wenn zum Beispiel eine AIR-Anwendung installiert wird, zeigt das AIR-Installationsprogramm die identifizierenden Informationen aus dem Zertifikat des Herausgebers an, wenn der Benutzer gefragt wird, ob die Anwendung installiert werden soll. In anderen Fällen müssen vielleicht die öffentlichen Schlüssel oder andere Zertifikatinformationen mit einer Liste akzeptierter Schlüssel verglichen werden. (Diese Liste muss gesichert sein, eventuell mit einer eigenen Signatur oder durch Speicherung im verschlüsselten lokalen Speicher von AIR, damit die Liste nicht ihrerseits manipuliert werden kann.)

Hinweis: *Sie können sich zwar auch ohne unabhängige Verifizierung dafür entscheiden, dem signierenden Zertifikat zu vertrauen – beispielsweise bei selbst signierten Zertifikaten – dadurch ist in puncto Sicherheit jedoch nicht viel gewonnen. Solange Sie nicht wissen, wer die Signatur erstellt hat, hat die Versicherung, dass die Signatur nicht manipuliert wurden, nur einen sehr geringen Wert. Bei der Signatur könnte es sich um eine gültig signierte Fälschung handeln.*

Gültigkeitsdauer und Sperre von Zertifikaten

Adobe AIR 1.5 und höher

Alle Zertifikat haben eine begrenzte Gültigkeitsdauer. Zertifikate können auch von der ausgebenden Zertifizierungsstelle gesperrt werden, zum Beispiel wenn der private Schlüssel des Zertifikats nicht mehr vertrauenswürdig ist oder gestohlen wurde. Wenn eine Signatur mit einem abgelaufenen oder gesperrten Zertifikat signiert ist, wird die Signatur als ungültig betrachtet, sofern nicht ein Zeitstempel als Teil der Signatur enthalten ist. Wenn ein Zeitstempel vorhanden ist, validiert die XMLSignatureValidator-Klasse die Signatur, falls das Zertifikat zum Zeitpunkt des Signierens gültig war.

Ein Zeitstempel ist eine signierte digitale Meldung von einem Zeitstempeldienst, der zertifiziert, dass die Daten zu einem bestimmten Zeitpunkt (Datum und Uhrzeit) signiert wurden. Zeitstempel werden von Zeitstempelstellen ausgegeben und durch das Zertifikat der Zeitstempelstelle signiert. Das in den Zeitstempel eingebettete Zertifikat der Zeitstempelstelle muss auf dem zurzeit verwendeten Computer als vertrauenswürdig gelten, damit der Zeitstempel als gültig anerkannt wird. Die XMLSignatureValidator-Klasse stellt keine API bereit, mit der ein anderes Zertifikat zur Verwendung bei der Zeitstempelvalidierung bestimmt wird.

Implementieren der IURIDereferencer-Schnittstelle

Adobe AIR 1.5 und höher

Für die Validierung einer XML-Signatur müssen Sie eine Implementierung der IURIDereferencer-Schnittstelle bereitstellen. Die Implementierung ist dafür zuständig, die URIs innerhalb der Reference-Elemente eines XML-Signaturdokuments aufzulösen und die Daten zurückzugeben, damit der Digest berechnet werden kann. Der berechnete Digest wird mit dem Digest in der Signatur verglichen, um festzustellen, ob die referenzierten Daten seit dem Erstellen der Signatur geändert wurden.

Hinweis: *HTML-basierte AIR-Anwendungen müssen eine SWF-Bibliothek importieren, die eine ActionScript-Implementierung enthält, um XML-Signaturen zu validieren. In JavaScript kann die IURIDereferencer-Schnittstelle nicht implementiert werden.*

Die IURIDerefencer-Schnittstelle verfügt über eine einzelne Methode, `dereference(uri:String)`, die implementiert werden muss. Das XMLSignatureValidator-Objekt ruft diese Methode für jeden Verweis in der Signatur aus. Die Methode muss die Daten in einem ByteArray-Objekt zurückgeben.

In den meisten Fällen müssen Sie auch Eigenschaften oder Methoden hinzufügen, mit denen das Dereferencer-Objekt die referenzierten Daten finden kann. Wenn sich die signierten Daten zum Beispiel in demselben Dokument wie die Signatur befinden, können Sie eine Mitgliedsvariable hinzufügen, die einen Verweis auf das XML-Dokument bereitstellt. Die `dereference()`-Methode kann diese Variable dann zusammen mit dem URI verwenden, um die referenzierten Daten zu suchen. Befinden sich die signierten Daten dagegen in einem Verzeichnis des lokalen Dateisystems, benötigt die `dereference()`-Methode möglicherweise eine Eigenschaft, die den Pfad zu diesem Verzeichnis angibt, damit die referenzierten Dateien gefunden werden.

Die XMLSignatureValidator-Klasse verlässt sich bei der Interpretation von URI-Strings vollständig auf den Dereferenzierer. Die Standardregeln für die Dereferenzierung von URIs sind in Abschnitt 4.3.3 der W3C-Empfehlung für die Syntax und Verarbeitung von XML-Signaturen aufgeführt.

Dereferenzieren von URIs in umhüllten Signaturen

Adobe AIR 1.5 und höher

Wenn eine umhüllte XML-Signatur generiert wird, werden die Signaturelemente in die signierten Daten eingefügt. Wenn Sie zum Beispiel die folgende Meldung mit einer umhüllten Signaturstruktur signieren:

```
<message>
    <data>...</data>
</message>
```

Dann sieht das signierte Dokument folgendermaßen aus:

```
<message>
    <data>...</data>
     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <Reference URI="">
                <Transforms>
                    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
                </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <DigestValue>yv6...Z0Y=</DigestValue>
            </Reference>
        </SignedInfo>
        <SignatureValue>cCY...LQ==</SignatureValue>
        <KeyInfo>
            <X509Data>
                <X509Certificate>MII...4e</X509Certificate>
            </X509Data>
        </KeyInfo>
    </Signature>
</message>
```

Beachten Sie, dass die Signatur ein einzelnes Reference-Element mit einem leeren String als URI enthält. Ein leerer String verweist in diesem Kontext auf den Stamm des Dokuments.

Beachten Sie auch, dass der Transformalgorithmus angibt, dass eine umhüllte Signaturtransformation angewendet wurde. Wenn eine umhüllte Signaturtransformationen angewendet wurde, entfernt der XMLSignatureValidator die Signatur automatisch aus dem Dokument, bevor der Digest berechnet wird. Dies bedeutet, dass der Dereferenzierer das Signature-Element nicht zu entfernen braucht, wenn die Daten zurückgegeben werden.

Im folgenden Beispiel wird ein Dereferenzierer für umhüllte Signaturen veranschaulicht:

```
package
{
    import flash.events.ErrorEvent;
    import flash.events.EventDispatcher;
    import flash.security.IURIDereferencer;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;

    public class EnvelopedDereferencer
        extends EventDispatcher implements IURIDereferencer
    {
        private var signedMessage:XML;

        public function EnvelopedDereferencer( signedMessage:XML )
        {
            this.signedMessage = signedMessage;
        }

        public function dereference( uri:String ):IDataInput
        {
            try
            {
```

```
                if( uri.length != 0 )
                {
                    throw( new Error("Unsupported signature type.") );
                }
                var data:ByteArray = new ByteArray();
                data.writeUTFBytes( signedMessage.toXMLString() );
                data.position = 0;
            }
            catch (e:Error)
                {
                var error:ErrorEvent =
                    new ErrorEvent("Ref error " + uri + " ", false, false, e.message);
                this.dispatchEvent(error);
                data = null;
                throw new Error("Reference not resolvable: " + uri + ", " + e.message);
            }
            finally
            {
                return data;
            }
        }
    }
}
```

Diese Dereferenziererklasse verwendet eine Konstruktorfunktion mit einem Parameter, `signedMessage`, um das umhüllte Signaturdokument der `dereference()`-Methode verfügbar zu machen. Da sich der Verweis in einer umhüllten Signatur immer auf den Stamm der signierten Daten bezieht, schreibt die `dereferencer()`-Methode das Dokument in ein ByteArray und gibt es zurück.

Dereferenzieren von URIs in umhüllenden und getrennten Signaturen

Adobe AIR 1.5 und höher

Wenn sich die signierten Daten in demselben Dokument wie die Signatur befinden, verwenden die URIs in den Verweisen normalerweise die XPath- oder XPointer-Syntax, um die signierten Elemente zu adressieren. Die W3C-Empfehlung für die Syntax und Verarbeitung von XML-Signaturen empfiehlt diese Syntax lediglich; deshalb sollte Ihre Implementierung auf den Signaturen basieren, die Sie erwarten (und Sie sollten ausreichende Fehlerüberprüfung hinzufügen, um eine nicht unterstützte Syntax definiert zu verarbeiten).

Die Signatur einer AIR-Anwendung ist ein Beispiel für eine umhüllende Signatur. Die Dateien der Anwendung sind in einem Manifest-Element aufgeführt. Das Manifest-Element wird im Reference URI-Attribut mit dem String „#PackageContents“, der auf die ID des Manifest-Elements verweist, adressiert:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="PackageSignature">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
        <SignatureMethod Algorithm="http://www.w3.org/TR/xmldsig-core#rsa-sha1"/>
        <Reference URI="#PackageContents">
            <Transforms>
                <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>ZMGqQdaRKQc1HirIRsDpeBDlaElS+pPotdziIAyAYDk=</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue Id="PackageSignatureValue">cQK...7Zg==</SignatureValue>
    <KeyInfo>
        <X509Data>
            <X509Certificate>MII...T4e</X509Certificate>
        </X509Data>
    </KeyInfo>
    <Object>
    <Manifest Id="PackageContents">
        <Reference URI="mimetype">
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">
            </DigestMethod>
            <DigestValue>0/oCb84THKMagtI0Dy0KogEu92TegdesqRr/clXct1c=</DigestValue>
        </Reference>
        <Reference URI="META-INF/AIR/application.xml">
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">
            </DigestMethod>
            <DigestValue>P9MqtqSdqcqnFgeoHCJysLQu4PmbUW2JdAnc1WLq8h4=</DigestValue>
        </Reference>
        <Reference URI="XMLSignatureValidation.swf">
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">
            </DigestMethod>
            <DigestValue>OliRHRAgc9qt3Dk0m0Bi53Ur5ur3fAweIFwju74rFgE=</DigestValue>
        </Reference>
    </Manifest>
</Object>
</Signature>
```

Ein Dereferenzierer zum Validieren dieser Signatur muss den URI-String, der `"#PackageContents"` enthält, aus dem Reference-Element nehmen und das Manifest-Element in einem ByteArray-Objekt zurückgeben. Das Symbol „#“ bezieht sich auf den Wert eines Element-ID-Attributs.

Im folgenden Beispiel wird ein Dereferenzierer für die Validierung der Signaturen von AIR-Anwendungen implementiert. Die Implementierung wird einfach gehalten, indem sie von der bekannten Struktur einer AIR-Signatur ausgeht. Ein Dereferenzierer für allgemeine Zwecke könnte erheblich komplexer sein.

```
package
{
    import flash.events.ErrorEvent;
    import flash.security.IURIDereferencer;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;

    public class AIRSignatureDereferencer implements IURIDereferencer {
        private const XML_SIG_NS:Namespace =
            new Namespace( "http://www.w3.org/2000/09/xmldsig#" );
        private var airSignature:XML;

        public function AIRSignatureDereferencer( airSignature:XML ) {
            this.airSignature = airSignature;
        }

        public function dereference( uri:String ):IDataInput {
            var data:ByteArray = null;
            try
            {
                if( uri != "#PackageContents" )
                {
                    throw( new Error("Unsupported signature type.") );
                }
                var manifest:XMLList =
                    airSignature.XML_SIG_NS::Object.XML_SIG_NS::Manifest;
                data = new ByteArray();
                data.writeUTFBytes( manifest.toXMLString());
                data.position = 0;
            }
            catch (e:Error)
            {
                data = null;
                throw new Error("Reference not resolvable: " + uri + ", " + e.message);
            }
            finally
            {
                return data;
            }

        }
    }
}
```

Wenn Sie diesen Signaturtyp überprüfen, werden nur die Daten im Manifest-Element validiert. Die eigentlichen Dateien im Paket werden in keiner Weise überprüft. Um zu prüfen, ob die Paketdateien manipuliert wurden, berechnen Sie den SHA256-Digest und vergleichen Sie das Ergebnis mit dem im Manifest verzeichneten Digest. Der XMLSignatureValidator überprüft derartige sekundäre Verweise nicht automatisch.

Hinweis: *Dieses Beispiel ist nur zur Veranschaulichung des Ablaufs einer Signaturvalidierung aufgeführt. Eine AIR-Anwendung, die ihre eigene Signatur validiert, ist nicht besonders sinnvoll. Wenn die Anwendung bereits manipuliert wurde, könnte auch einfach die Validierungsprüfung entfernt werden.*

Berechnen von Digestwerten für externe Ressourcen

Adobe AIR 1.5 und höher

AIR enthält keine integrierten Funktionen für die Berechnung von SHA256-Digests, das Flex SDK enthält jedoch eine SHA256-Utility-Klasse. Das SDK enthält auch die Base64-Encoder-Utility-Klasse, die hilfreich beim Vergleichen des berechneten Digests mit dem in der Signatur gespeicherten Digest ist.

Die folgende Beispielfunktion liest und validiert die Dateien in einem AIR-Paketmanifest:

```
import mx.utils.Base64Encoder;
import mx.utils.SHA256;

private function verifyManifest( sigFile:File, manifest:XML ):Boolean
{
    var result:Boolean = true;
    var message:String = '';
    var nameSpace:Namespace = manifest.namespace();

    if( manifest.nameSpace::Reference.length() <= 0 )
    {
        result = false;
        message = "Nothing to validate.";
    }
    for each (var reference:XML in manifest.nameSpace::Reference)
    {
        var file:File = sigFile.parent.parent.resolvePath( reference.@URI );
        var stream:FileStream = new FileStream();
        stream.open(file, FileMode.READ);
        var fileData:ByteArray = new ByteArray();
        stream.readBytes( fileData, 0, stream.bytesAvailable );

        var digestHex:String = SHA256.computeDigest( fileData );
        //Convert hexidecimal string to byte array
        var digest:ByteArray = new ByteArray();
        for( var c:int = 0; c < digestHex.length; c += 2 ){
            var byteChar:String = digestHex.charAt(c) + digestHex.charAt(c+1);
            digest.writeByte( parseInt( byteChar, 16 ));
        }
        digest.position = 0;
```

```
        var base64Encoder:Base64Encoder = new Base64Encoder();
        base64Encoder.insertNewLines = false;
        base64Encoder.encodeBytes( digest, 0, digest.bytesAvailable );
        var digestBase64:String = base64Encoder.toString();
        if( digestBase64 == reference.nameSpace::DigestValue )
        {
            result = result && true;
            message += "   " + reference.@URI + " verified.\n";
        }
        else
        {
            result = false;
            message += " ---- " + reference.@URI + " has been modified!\n";
        }
        base64Encoder.reset();
    }
    trace( message );
    return result;
}
```

Die Funktion durchläuft alle Verweise im Manifest-Element. Für jeden Verweis wird der SHA256-Digest berechnet, im Base64-Format kodiert und mit dem Digest im Manifest verglichen. Die URIs in einem AIR-Paket verweisen auf Pfade relativ zum Anwendungsverzeichnis. Die Pfade werden basierend auf dem Speicherort der Signaturdatei aufgelöst, die sich immer im META-INF-Unterverzeichnis im Anwendungsverzeichnis befindet. Beachten Sie, dass die Flex-SHA256-Klasse den Digest als String von hexadezimalen Zeichen zurückgibt. Dieser String muss in ein ByteArray konvertiert werden, das die vom hexadezimalen String repräsentierten Byte enthält.

Um die mx.utils.SHA256- und Base64Encoder-Klassen in Flash CS4 zu verwenden, können Sie diese Klassen entweder suchen und in Ihr Verzeichnis für die Anwendungsentwicklung kopieren oder mit dem Flex SDK eine Bibliotheks-SWF-Datei kompilieren, die diese Klassen enthält.

Dereferenzieren von URIs in separaten Signaturen, die externe Daten referenzieren

Adobe AIR 1.5 und höher

Wenn ein URI auf eine externe Ressource verweist, muss auf die Daten zugegriffen werden und sie müssen in ein ByteArray-Objekt geladen werden. Wenn der URI eine absolute URL enthält, muss einfach nur eine Datei gelesen oder eine URL angefordert werden. Wenn der URI einen relativen Pfad enthält, was weitaus häufiger der Fall ist, muss Ihre IURIDereferencer-Implementierung eine Möglichkeit zum Auflösen der Pfade zu den signierten Dateien enthalten.

Im folgenden Beispiel wird ein File-Objekt initialisiert, wenn die Dereferenziererinstanz als Basis zum Auflösen signierter Dateien aufgebaut ist.

```
package
{
    import flash.events.ErrorEvent;
    import flash.events.EventDispatcher;
    import flash.filesystem.File;
    import flash.filesystem.FileMode;
    import flash.filesystem.FileStream;
    import flash.security.IURIDereferencer;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;
    public class RelativeFileDereferencer
        extends EventDispatcher implements IURIDereferencer
    {
        private var base:File;

        public function RelativeFileDereferencer( base:File )
        {
            this.base = base;
        }

        public function dereference( uri:String ):IDataInput
        {
            var data:ByteArray = null;
            try{
                var referent:File = this.base.resolvePath( uri );
                var refStream:FileStream = new FileStream();
                data = new ByteArray();
                refStream.open( referent, FileMode.READ );

                refStream.readBytes( data, 0, data.bytesAvailable );

            } catch ( e:Error ) {
                data = null;
                throw new Error("Reference not resolvable: " + referent.nativePath + ", " +
e.message );
            } finally {
                return data;
            }
        }
    }
}
```

Die `dereference()`-Funktion sucht die Datei, die vom Referenz-URI adressiert wird, lädt den Dateiinhalt in ein Byte-Array und gibt ein ByteArray-Objekt zurück.

Hinweis: *Vor dem Validieren remoter externer Verweise sollten Sie überlegen, ob Ihre Anwendung durch ein auf schädigende Weise konstruiertes Signaturdokument mit einem „Call home"-Angriff (durch Spyware) oder ähnlichem geschädigt werden kann.*

Kapitel 49: Clientsystem-Umgebung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In Folgenden wird die Interaktion mit dem System des Benutzers erläutert. Der Abschnitt enthält Informationen zum Ermitteln der unterstützten Funktionen und zum Erstellen mehrsprachiger Anwendungen mit dem installierten Eingabemethoden-Editor (IME, Input Method Editor) des Benutzers (sofern verfügbar). Darüber hinaus werden typische Einsatzmöglichkeiten von Anwendungsdomänen erläutert.

Verwandte Hilfethemen

flash.system.System

flash.system.Capabilities

Grundlagen der Clientsystem-Umgebung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Erstellen komplexerer Anwendungen benötigen Sie möglicherweise detaillierte Informationen zum Betriebssystem des Benutzers; eventuell müssen Sie auch auf die Funktionen dieses Betriebssystems zugreifen können. Das flash.system-Paket enthält mehrere Klassen, mit denen Sie auf Systemfunktionalität zugreifen können, wie zum Beispiel:

- Ermitteln der Anwendung und der Sicherheitsdomäne, in denen Code ausgeführt wird.
- Ermitteln der Leistungsmerkmale der vom Benutzer verwendeten Instanz der Flash-Laufzeitumgebung (wie Flash® Player oder Adobe® AIR™), wie zum Beispiel Bildschirmgröße (Auflösung) und Verfügbarkeit bestimmter Funktionen, z. B. MP3-Audio.
- Erstellen mehrsprachiger Websites unter Verwendung des IME
- Interagieren mit dem Container der Flash-Laufzeitumgebung (z. B. eine HTML-Seite oder eine Containeranwendung).
- Speichern von Daten in der Zwischenablage des Benutzers

Das flash.system-Paket enthält zudem die Klassen „IMEConversionMode“ und „SecurityPanel“. Diese Klassen enthalten statische Konstanten, die jeweils mit den Klassen „IME“ und „Security“ verwendet werden.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe aufgeführt:

Betriebssystem Das Hauptprogramm, das auf einem Computer ausgeführt wird und über das alle anderen Anwendungen ausgeführt werden, z. B. Microsoft Windows, Mac OS X oder Linux®.

Clipboard Der Container des Betriebssystems zum Speichern von Text oder Objekten, die kopiert oder ausgeschnitten werden und die in Anwendungen eingefügt werden können.

Anwendungsdomäne Ein Mechanismus zum Trennen von Klassen in verschiedenen SWF-Dateien, sodass die einzelnen Klassen nicht überschrieben werden, wenn SWF-Dateien unterschiedliche Klassen mit demselben Namen enthalten.

IME (Eingabemethoden-Editor) Ein Programm (oder Tool des Betriebssystems), mit dem komplexe Zeichen oder Symbole über eine Standardtastatur eingegeben werden können.

Clientsystem In der Programmierterminologie handelt es sich bei einem Client um den Teil der Anwendung (oder um die gesamte Anwendung), die auf dem Computer eines Benutzers ausgeführt und von einem einzelnen Benutzer verwendet wird. Das Clientsystem ist das zugrunde liegende Betriebssystem auf dem Computer des Benutzers.

Verwenden der System-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die System-Klasse enthält Methoden und Eigenschaften, mit denen Sie mit dem Betriebssystem des Benutzers interagieren und die aktuelle Speicherauslastung der Laufzeit abrufen können. Mit den Methoden und Eigenschaften der System-Klasse können Sie zudem `imeComposition`-Ereignisse überwachen, die Laufzeit so konfigurieren, dass externe Textdateien über die aktuelle Codepage des Benutzers oder im Unicode-Format geladen werden, oder den Inhalt der Zwischenablage des Benutzers festlegen.

Abrufen der Informationen zum System des Benutzers zur Laufzeit

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Durch Überprüfen der `System.totalMemory`-Eigenschaft können Sie den Speicherplatz (in Byte) ermitteln, den die Laufzeit derzeit belegt. Mit dieser Eigenschaft können Sie die Speicherauslastung überwachen und die Anwendungen entsprechend den Änderungen der Speichernutzung optimieren. Wenn die Speicherauslastung beispielsweise aufgrund eines bestimmten visuellen Effekts erheblich zunimmt, empfiehlt es sich unter Umständen, den Effekt zu ändern oder zu entfernen.

Die `System.ime`-Eigenschaft verweist auf den aktuellen installierten Eingabemethoden-Editor (IME, Input Method Editor). Mit dieser Eigenschaft können Sie `imeComposition`-Ereignisse (`flash.events.IMEEvent.IME_COMPOSITION`) über die `addEventListener()`-Methode überwachen.

Die dritte Eigenschaft der System-Klasse ist die `useCodePage`-Eigenschaft. Wenn `useCodePage` auf `true` gesetzt ist, verwendet die Laufzeit die Standardcodepage des Betriebssystems, um externe Textdateien zu laden. Wenn Sie diese Eigenschaft auf `false` setzen, werden externe Dateien in der Laufzeit als Unicode interpretiert.

Wenn Sie `System.useCodePage` auf `true` setzen, achten Sie darauf, dass die Standardcodepage des Betriebssystems den Zeichensatz der externen Textdatei enthält, damit der Text angezeigt werden kann. Wenn Sie beispielsweise eine externe Textdatei mit chinesischen Zeichen laden, werden die Zeichen unter einem System mit der Codepage für das englische Windows-Betriebssystem nicht angezeigt, da diese Codepage keine chinesischen Zeichen enthält.

Sie können sicherstellen, dass die externen Textdateien Ihrer Anwendung auf allen Plattformen angezeigt werden, indem Sie alle externen Textdateien als Unicode kodieren und für `System.useCodePage` die Standardeinstellung `false` beibehalten. Auf diese Weise interpretiert die Laufzeit den Text als Unicode.

Speichern von Text in der Zwischenablage

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die System-Klasse enthält die `setClipboard()`-Methode, über die in der Flash-Laufzeitumgebung die Zwischenablage des Benutzers mit dem angegebenen String gefüllt werden kann. Aus Sicherheitsgründen ist keine `Security.getClipboard()`-Methode vorhanden, da durch diese Methode böswilligen Websites unter Umständen der Zugriff auf die zuletzt in die Zwischenablage des Benutzers kopierten Daten ermöglicht wird.

Mit dem folgenden Code wird veranschaulicht, wie beim Auftreten eines Sicherheitsfehlers eine Fehlermeldung in die Zwischenablage des Benutzers kopiert werden kann. Die Fehlermeldung kann nützlich sein, wenn der Benutzer einen potenziellen Bug in einer Anwendung melden möchte.

```
private function securityErrorHandler(event:SecurityErrorEvent):void
{
    var errorString:String = "[" + event.type + "] " + event.text;
    trace(errorString);
    System.setClipboard(errorString);
}
```

Flash Player 10 und AIR 1.0

Sie können die Clipboard-Klasse verwenden, um Daten als Reaktion auf ein Benutzerereignis in die Zwischenablage zu schreiben oder aus der Zwischenablage zu lesen. In AIR ist kein Benutzerereignis erforderlich, damit Code, der in der Anwendungs-Sandbox ausgeführt wird, auf die Zwischenablage zugreifen kann.

Verwenden der Capabilities-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Capabilities-Klasse können Entwickler die Umgebung ermitteln, in der eine Anwendung ausgeführt wird. Mithilfe der verschiedenen Eigenschaften der Capabilities-Klasse können Sie die Bildschirmauflösung und die Sprache des Betriebssystems des Benutzers und die aktuell installierte Version der Flash-Laufzeitumgebung herausfinden sowie ermitteln, ob das jeweilige System barrierefreie Software unterstützt.

Durch Überprüfen der Eigenschaften der Capabilities-Klasse können Sie die Anwendung so anpassen, dass sie optimal mit der spezifischen Umgebung des Benutzers eingesetzt werden kann. Wenn Sie beispielsweise die Eigenschaften `Capabilities.screenResolutionX` und `Capabilities.screenResolutionY` überprüfen, können Sie die Anzeigeauflösung des Systems ermitteln und die geeignete Videogröße festlegen. Durch Überprüfen der `Capabilities.hasMP3`-Eigenschaft können Sie feststellen, ob die MP3-Wiedergabe auf dem System des Benutzers unterstützt wird, bevor Sie eine externe MP3-Datei laden.

Im folgenden Code wird mithilfe eines regulären Ausdrucks die Version der Flash-Laufzeitumgebung analysiert, die der Client verwendet:

```
var versionString:String = Capabilities.version;
var pattern:RegExp = /^(\w*) (\d*),(\d*),(\d*),(\d*)$/;
var result:Object = pattern.exec(versionString);
if (result != null)
{
    trace("input: " + result.input);
    trace("platform: " + result[1]);
    trace("majorVersion: " + result[2]);
    trace("minorVersion: " + result[3]);
    trace("buildNumber: " + result[4]);
    trace("internalBuildNumber: " + result[5]);
}
else
{
    trace("Unable to match RegExp.");
}
```

Wenn Sie die Systemangaben des Benutzers an ein Serverskript senden möchten, um die Daten in einer Datenbank zu speichern, verwenden Sie den folgenden ActionScript-Code:

```
var url:String = "log_visitor.cfm";
var request:URLRequest = new URLRequest(url);
request.method = URLRequestMethod.POST;
request.data = new URLVariables(Capabilities.serverString);
var loader:URLLoader = new URLLoader(request);
```

Capabilities-Beispiel: Ermitteln von Systemeigenschaften

Flash Player 9 und höher

Mit der Beispielanwendung „CapabilitiesExplorer“ wird veranschaulicht, wie Sie mithilfe der flash.system.Capabilities-Klasse die Funktionen ermitteln können, die die Version der Flash-Laufzeitumgebung des Benutzers unterstützt. In diesem Beispiel werden die folgenden Verfahren vermittelt:

- Ermitteln der in der Flash-Laufzeitumgebungsversion des Benutzers unterstützten Funktionen mithilfe der Capabilities-Klasse
- Verwenden der ExternalInterface-Klasse zum Ermitteln der im Browser des Benutzers unterstützten Einstellungen

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien der Anwendung „CapabilitiesExplorer“ befinden sich im Ordner „Samples/CapabilitiesExplorer“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
CapabilitiesExplorer.fla oder CapabilitiesExplorer.mxml	Die Hauptanwendungsdatei im Flash-Format (FLA) oder Flex-Format (MXML).
com/example/programmingas3/capabilities/CapabilitiesGrabber.as	Die Klasse mit den Hauptfunktionen der Anwendung, einschließlich Hinzufügen der Systeminformationen zu einem Array, Sortieren der Elemente und Abrufen der Browserinformationen mit der ExternalInterface-Klasse.
capabilities.html	Ein HTML-Container mit dem für die Verbindung mit der externen API erforderlichen JavaScript-Code.

Überblick über die Anwendung „CapabilitiesExplorer“

Flash Player 9 und höher

Die Datei „CapabilitiesExplorer.mxml“ dient zum Einrichten der Benutzeroberfläche für die Anwendung „CapabilitiesExplorer“. Die Eigenschaften der Flash-Laufzeitumgebungsversion des Benutzers werden in einer DataGrid-Komponenteninstanz auf der Bühne angezeigt. Wenn die Anwendung über einen HTML-Container ausgeführt wird und die externe API verfügbar ist, werden auch die Browserinformationen angezeigt.

Wenn das `creationComplete`-Ereignis in der Datei der Hauptanwendung ausgelöst wurde, wird die `initApp()`-Methode aufgerufen. Mit der `initApp()`-Methode wird die `getCapabilities()`-Methode über die com.example.programmingas3.capabilities.CapabilitiesGrabber-Klasse aufgerufen. Der Code für die `initApp()`-Methode lautet wie folgt:

```
private function initApp():void
{
    var dp:Array = CapabilitiesGrabber.getCapabilities();
    capabilitiesGrid.dataProvider = dp;
}
```

Die `CapabilitiesGrabber.getCapabilities()`-Methode gibt ein sortiertes Array mit Informationen zur Flash-Laufzeitumgebung und zum Browser zurück, das dann an die `dataProvider`-Eigenschaft der DataGrid-Komponenteninstanz `capabilitiesGrid` auf der Bühne übergeben wird.

Überblick über die CapabilitiesGrabber-Klasse

Flash Player 9 und höher

Mit der statischen `getCapabilities()`-Methode der CapabilitiesGrabber-Klasse werden alle Eigenschaften der flash.system.Capabilities-Klasse zu einem Array (`capDP`) hinzugefügt. Dann wird die statische `getBrowserObjects()`-Methode der CapabilitiesGrabber-Klasse aufgerufen. Bei der `getBrowserObjects()`-Methode wird mithilfe der externen API das Navigatorobjekt des Browsers durchlaufen, das die Browserinformationen enthält. Die `getCapabilities()`-Methode lautet wie folgt:

```
public static function getCapabilities():Array
{
    var capDP:Array = new Array();
    capDP.push({name:"Capabilities.avHardwareDisable", value:Capabilities.avHardwareDisable});
    capDP.push({name:"Capabilities.hasAccessibility", value:Capabilities.hasAccessibility});
    capDP.push({name:"Capabilities.hasAudio", value:Capabilities.hasAudio});
    ...
    capDP.push({name:"Capabilities.version", value:Capabilities.version});
    var navArr:Array = CapabilitiesGrabber.getBrowserObjects();
    if (navArr.length > 0)
    {
        capDP = capDP.concat(navArr);
    }
    capDP.sortOn("name", Array.CASEINSENSITIVE);
    return capDP;
}
```

Die `getBrowserObjects()`-Methode gibt ein Array aller Eigenschaften im Navigatorobjekt des Browsers zurück. Wenn dieses Array mindestens ein Element aufweist, wird das Array der Browserinformationen (`navArr`) an das Array der Flash Player-Informationen (`capDP`) angehängt. Das gesamte Array wird dann alphabetisch sortiert. Das sortierte Array wird schließlich an die Datei der Hauptanwendung zurückgegeben, die dann das Datenraster füllt. Der Code für die `getBrowserObjects()`-Methode lautet wie folgt:

```
private static function getBrowserObjects():Array
{
    var itemArr:Array = new Array();
    var itemVars:URLVariables;
    if (ExternalInterface.available)
    {
        try
        {
            var tempStr:String = ExternalInterface.call("JS_getBrowserObjects");
            itemVars = new URLVariables(tempStr);
            for (var i:String in itemVars)
            {
                itemArr.push({name:i, value:itemVars[i]});
            }
        }
        catch (error:SecurityError)
        {
            // ignore
        }
    }
    return itemArr;
}
```

Wenn die externe API in der aktuellen Umgebung des Benutzers verfügbar ist, wird in der Flash-Laufzeitumgebung die JavaScript-Methode `JS_getBrowserObjects()` aufgerufen, mit der dann das Navigatorobjekt des Browsers durchlaufen und ein String mit URL-kodierten Werten an ActionScript zurückgegeben wird. Dieser String wird dann in ein URLVariables-Objekt umgewandelt (`itemVars`) und zum `itemArr`-Array hinzugefügt, das an das aufrufende Skript zurückgegeben wird.

Kommunikation mit JavaScript

Flash Player 9 und höher

Bei der Erstellung der Anwendung „CapabilitiesExplorer“ muss abschließend der nötige JavaScript-Code geschrieben werden, mit dem die einzelnen Elemente im Navigatorobjekt des Browsers durchlaufen werden sowie ein Name-Wert-Paar an ein temporäres Array angehängt wird. Der Code für die JavaScript-Methode `JS_getBrowserObjects()` in der Datei „container.html“ lautet wie folgt:

```
<script language="JavaScript">
    function JS_getBrowserObjects()
    {
        // Create an array to hold each of the browser's items.
        var tempArr = new Array();

        // Loop over each item in the browser's navigator object.
        for (var name in navigator)
        {
            var value = navigator[name];

            // If the current value is a string or Boolean object, add it to the
            // array, otherwise ignore the item.
            switch (typeof(value))
            {
                case "string":
                case "boolean":

                    // Create a temporary string which will be added to the array.
                    // Make sure that we URL-encode the values using JavaScript's
                    // escape() function.
                    var tempStr = "navigator." + name + "=" + escape(value);
                    // Push the URL-encoded name/value pair onto the array.
                    tempArr.push(tempStr);
                    break;
            }
        }
        // Loop over each item in the browser's screen object.
        for (var name in screen)
        {
            var value = screen[name];

            // If the current value is a number, add it to the array, otherwise
            // ignore the item.
            switch (typeof(value))
            {
                case "number":
                    var tempStr = "screen." + name + "=" + escape(value);
                    tempArr.push(tempStr);
                    break;
            }
        }
        // Return the array as a URL-encoded string of name-value pairs.
        return tempArr.join("&");
    }
</script>
```

Im Code wird zunächst ein temporäres Array erstellt, das alle Name-Wert-Paare des Navigatorobjekts enthält. Als Nächstes wird das Navigatorobjekt mit der `for..in`-Schleife durchlaufen und der Datentyp des aktuellen Werts geprüft, um unerwünschte Werte zu filtern. In dieser Anwendung werden nur String- oder Boolean-Werte berücksichtigt. Andere Datentypen (z. B. Funktionen oder Arrays) werden dagegen außer Acht gelassen. Alle String- oder Boolean-Werte im Navigatorobjekt werden an das `tempArr`-Array angehängt. Als Nächstes wird das Bildschirmobjekt des Browsers mit einer `for..in`-Schleife durchlaufen und jeder numerische Wert wird zum `tempArr`-Array hinzugefügt. Schließlich wird das temporäre Array mit der `Array.join()`-Methode in einen String umgewandelt. Im Array werden Und-Zeichen (&) als Trennzeichen verwendet. Dadurch können die Daten in ActionScript mit der URLVariables-Klasse analysiert werden.

Kapitel 50: Aufrufen und Beenden von AIR-Anwendungen

Adobe AIR 1.0 und höher

In diesem Abschnitt finden Sie Informationen zum Aufrufen einer installierten Adobe® AIR®-Anwendung sowie zu Optionen und Überlegungen zum Schließen einer Anwendung, die gerade ausgeführt wird.

Hinweis: *Die NativeApplication-, InvokeEvent- und BrowserInvokeEvent-Objekte sind nur für SWF-Inhalt verfügbar, der in der AIR-Anwendungs-Sandbox ausgeführt wird. SWF-Inhalt, der in der Flash Player-Laufzeitumgebung, im Browser oder Standalone-Player (Projektor), oder in einer AIR-Anwendung außerhalb der Anwendungs-Sandbox ausgeführt wird, kann nicht auf diese Klassen zugreifen.*

Eine kurze Erklärung sowie Codebeispiele zum Aufrufen und Beenden von AIR-Anwendungen finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Startup Options

Verwandte Hilfethemen

flash.desktop.NativeApplication

flash.events.InvokeEvent

flash.events.BrowserInvokeEvent

Aufrufen der Anwendung

Adobe AIR 1.0 und höher

AIR-Anwendungen werden durch die folgenden Aktionen des Benutzers (oder des Betriebssystems) aufgerufen:

- Start des Programms aus der Desktop-Shell.
- Verwendung der Anwendung als Befehl in einer Befehlszeilen-Shell.
- Öffnen eines Dateityps, dem die Anwendung als Standardanwendung zugeordnet wurde.
- (Mac OS X) Klicken auf das Anwendungssymbol in der Dock-Taskleiste (unabhängig davon, ob die Anwendung läuft oder nicht).
- Aufruf der Anwendung aus dem Installationsprogramm (nach Abschluss einer Neuinstallation oder nach Doppelklick auf der AIR-Datei einer bereits installierten Anwendung).
- Start einer Aktualisierung der AIR-Anwendung, wenn die installierte Version einen Hinweis eingeblendet hat, dass Aktualisierungen von der Anwendung selbst ausgeführt werden (durch Aufnahme der Deklaration `<customUpdateUI>true</customUpdateUI>` in die Anwendungsdeskriptordatei).
- (iOS) Empfängt eine Benachrichtigung vom Apple Push Notification service (APNs).
- Ruft die Anwendung über eine URL auf.

- Besuch einer Internetseite, die eine Flash-Badge oder eine Anwendung enthält, welche die Methode `com.adobe.air.AIR launchApplication()` aufrufen, die Identifikationsinformationen für die AIR-Anwendung enthält. (Der Anwendungsdeskriptor muss auch die Deklaration `<allowBrowserInvocation>true</allowBrowserInvocation>` enthalten, damit der Browser erfolgreich aufgerufen werden kann.)

Bei jedem Aufruf einer AIR-Anwendung löst AIR ein InvokeEvent-Objekt des Typs `invoke` durch das Singleton-Objekt NativeApplication aus. Um der Anwendung die für die Initialisierung und die Registrierung eines Ereignis-Listeners benötigte Zeit zu geben, werden `invoke`-Ereignisse in eine Warteschlange eingereiht und nicht verworfen. Sobald ein Listener registriert ist, werden die Ereignisse in der Warteschlange übermittelt.

Hinweis: *Wird die Anwendung durch die Browseraufruffunktion aufgerufen, löst das NativeApplication-Objekt nur dann ein `invoke`-Ereignis aus, wenn die Anwendung nicht bereits läuft.*

Um `invoke`-Ereignisse zu erhalten, rufen Sie die Methode `addEventListener()` des NativeApplication-Objekts (`NativeApplication.nativeApplication`) auf. Wenn sich ein Ereignis-Listener für ein `invoke`-Ereignis registriert, empfängt es auch alle `invoke`-Ereignisse, die vor der Registrierung aufgetreten sind. Nachdem der Aufruf an `addEventListener()` zurückgegeben wurde, werden die `invoke`-Ereignisse in der Warteliste in kurzen Abständen aufeinander folgend ausgelöst. Tritt während dieses Vorgangs ein neues `invoke`-Ereignis auf, kann es vor anderen Ereignissen in der Warteliste ausgelöst werden. Durch diese Einreihung von Ereignissen in eine Warteliste können Sie `invoke`-Ereignisse, die vor der Ausführung des Initialisierungscodes auftraten, verarbeiten. Denken Sie beim Hinzufügen eines Ereignis-Listeners zu einem späteren Zeitpunkt in der Ausführung (nach der Initialisierung der Anwendung) daran, dass dieser immer noch alle `invoke`-Ereignisse empfängt, die seit dem Starten der Anwendung aufgetreten sind.

Es wird nur eine Instanz der AIR-Anwendung geöffnet. Wird eine bereits laufende Anwendung erneut aufgerufen, löst AIR ein neues `invoke`-Ereignis an die bereits laufende Instanz aus. Es ist die Aufgabe der AIR-Anwendung, auf `invoke`-Ereignisse zu reagieren und die entsprechenden Schritte auszuführen (also zum Beispiel ein neues Dokumentfenster zu öffnen).

Ein `InvokeEvent`-Objekt enthält die Argumente, die an die Anwendung übergeben werden, sowie das Verzeichnis, aus dem die Anwendung aufgerufen wurde. Wenn die Anwendung aufgrund einer Dateityp-Verknüpfung aufgerufen wurde, dann wird der vollständige Pfad zur Datei in die Befehlszeilenargumente aufgenommen. Wurde die Anwendung wegen einer Anwendungsaktualisierung aufgerufen, wird der vollständige Pfad der AIR-Aktualisierungsdatei angegeben.

Wenn mehrere Dateien in einer Operation geöffnet werden, wird unter Mac OS X ein einzelnes InvokeEvent-Objekt ausgelöst. Jede Datei ist im `arguments`-Array enthalten. Unter Windows und Linux wird für jede Datei ein separates InvokeEvent-Objekt ausgelöst.

Die Anwendung kann `invoke`-Ereignisse durch die Registrierung eines Listeners über das NativeApplication-Objekt verarbeiten:

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvokeEvent);
```

Und durch die Definition eines Event-Listeners:

```
 var arguments:Array;
var currentDir:File;
public function onInvokeEvent(invocation:InvokeEvent):void {
    arguments = invocation.arguments;
    currentDir = invocation.currentDirectory;
}
```

Erfassen von Befehlszeilenargumenten

Adobe AIR 1.0 und höher

Die mit dem Aufruf einer AIR-Anwendung verknüpften Befehlszeilenargumente werden mit dem InvokeEvent-Objekt, das durch das NativeApplication-Objekt ausgelöst wurde, übermittelt. Die `arguments`-Eigenschaft von InvokeEvent enthält ein Array der Argumente, die beim Aufruf einer AIR-Anwendung vom Betriebssystem übergeben wurden. Enthalten diese Argumente relative Dateipfadangaben, können Sie die Pfadangaben in der Regel mit der Eigenschaft `currentDirectory` auflösen.

Die an eine AIR-Anwendung übergebenen Argumente werden wie durch Leerzeichen getrennte Strings behandelt, es sei denn, sie sind von doppelten Anführungszeichen eingeschlossen:

Argumente	Array
tick tock	{tick,tock}
tick "tick tock"	{tick,tick tock}
"tick" "tock"	{tick,tock}
\"tick\" \"tock\"	{"tick","tock"}

Die `currentDirectory`-Eigenschaft eines InvokeEvent-Objekts enthält ein File-Objekt, welches das Verzeichnis darstellt, aus dem die Anwendung gestartet wurde.

Wird eine Anwendung aufgerufen, weil eine Datei eines Typs geöffnet wird, der von der Anwendung registriert wurde, wird der native Pfad zur Datei in die Befehlszeilenargumente als String aufgenommen. (Ihre Anwendung ist für das Öffnen der Datei oder die Ausführung der geplanten Aktion an der Datei zuständig.) Wurde eine Anwendung so programmiert, dass Aktualisierungen automatisch durchgeführt werden (und nicht über die Standardbenutzeroberfläche für AIR-Aktualisierungen), wird der native Pfad zur AIR-Datei aufgenommen, wenn ein Benutzer auf eine AIR-Datei klickt, die eine Anwendung mit übereinstimmender Anwendungs-ID enthält.

Sie können mit der Methode `resolve()` des File-Objekts `currentDirectory` auf die Datei zugreifen:

```
 if((invokeEvent.currentDirectory != null)&&(invokeEvent.arguments.length > 0)){
    dir = invokeEvent.currentDirectory;
    fileToOpen = dir.resolvePath(invokeEvent.arguments[0]);
}
```

Überprüfen Sie, ob ein Argument tatsächlich den Pfad zu einer Datei angibt.

Beispiel: Aufruf eines Ereignisprotokolls

Adobe AIR 1.0 und höher

Im folgenden Beispiel wird gezeigt, wie Listener für das Ereignis `invoke` registriert werden und dieses verarbeiten. Im Beispiel werden alle eingehenden Aufrufereignisse protokolliert und das aktuelle Verzeichnis und die Befehlszeilenargumente angezeigt.

ActionScript-Beispiel

```
package
{
    import flash.display.Sprite;
    import flash.events.InvokeEvent;
    import flash.desktop.NativeApplication;
    import flash.text.TextField;

    public class InvokeEventLogExample extends Sprite
    {
        public var log:TextField;

        public function InvokeEventLogExample()
        {
            log = new TextField();
            log.x = 15;
            log.y = 15;
            log.width = 520;
            log.height = 370;
            log.background = true;

            addChild(log);

            NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvoke);
        }

        public function onInvoke(invokeEvent:InvokeEvent):void
        {
            var now:String = new Date().toTimeString();
            logEvent("Invoke event received: " + now);

            if (invokeEvent.currentDirectory != null)
            {
                logEvent("Current directory=" + invokeEvent.currentDirectory.nativePath);
            }
            else
            {
```

```
                logEvent("--no directory information available--");
            }

            if (invokeEvent.arguments.length > 0)
            {
                logEvent("Arguments: " + invokeEvent.arguments.toString());
            }
            else
            {
                logEvent("--no arguments--");
            }
        }

        public function logEvent(entry:String):void
        {
            log.appendText(entry + "\n");
            trace(entry);
        }
    }
}
```

Flex-Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
    invoke="onInvoke(event)" title="Invocation Event Log">
    <mx:Script>
    <![CDATA[
    import flash.events.InvokeEvent;
    import flash.desktop.NativeApplication;

    public function onInvoke(invokeEvent:InvokeEvent):void {
        var now:String = new Date().toTimeString();
        logEvent("Invoke event received: " + now);

        if (invokeEvent.currentDirectory != null){
            logEvent("Current directory=" + invokeEvent.currentDirectory.nativePath);
        } else {
            logEvent("--no directory information available--");
        }

        if (invokeEvent.arguments.length > 0){
            logEvent("Arguments: " + invokeEvent.arguments.toString());
        } else {
            logEvent("--no arguments--");
        }
    }

    public function logEvent(entry:String):void {
        log.text += entry + "\n";
        trace(entry);
    }
    ]]>
    </mx:Script>
    <mx:TextArea id="log" width="100%" height="100%" editable="false"
        valueCommit="log.verticalScrollPosition=log.textHeight;"/>
</mx:WindowedApplication>
```

Aufrufen einer AIR-Anwendung bei der Benutzeranmeldung

Adobe AIR 1.0 und höher

Durch die Einstellung der `startAtLogin`-Eigenschaft von NativeApplication auf `true` kann die AIR-Anwendung so eingerichtet werden, dass sie automatisch gestartet wird, wenn sich der Benutzer anmeldet. Sobald die Einstellung vorgenommen wurde, wird die Anwendung bei jeder Benutzeranmeldung automatisch gestartet. Dieser automatische Anwendungsaufruf beim Anmelden erfolgt solange, bis die Einstellung auf `false` gesetzt wird, der Benutzer die Einstellung manuell über das Betriebssystem ändert oder die Anwendung deinstalliert wird. Hierbei handelt es sich um eine Laufzeiteinstellung. Die Einstellung gilt nur für den aktuellen Benutzer. Die Anwendung muss installiert sein, damit die `startAtLogin`-Eigenschaft erfolgreich auf `true` gesetzt werden kann. Wird die Eigenschaft eingestellt, obwohl keine Anwendung installiert ist (wenn sie z. B. mit ADL gestartet wird), wird ein Fehler ausgegeben.

Hinweis: *Die Anwendung startet nicht beim Systemstart des Computers, sondern bei der Anmeldung des Benutzers.*

Um festzustellen, ob eine Anwendung automatisch oder durch eine Benutzeraktion aufgerufen wurde, können Sie die `reason`-Eigenschaft des InvokeEvent-Objekts untersuchen. Wenn die Eigenschaft gleich `InvokeEventReason.LOGIN` ist, wurde die Anwendung automatisch gestartet. Bei anderen Aufrufpfaden ist die `reason`-Eigenschaft wie folgt eingestellt:

- `InvokeEventReason.NOTIFICATION` (nur iOS) - Die Anwendung wurde über APNs aufgerufen. Weitere Informationen zum APNs finden Sie unter Verwenden von Push-Benachrichtigungen.
- `InvokeEventReason.OPEN_URL` - Die Anwendung wurde von einer anderen Anwendung oder vom System aufgerufen.
- `InvokeEventReason.Standard` - Alle anderen Fälle.

Damit Sie Zugriff auf die `reason`-Eigenschaft haben, muss Ihre Anwendung AIR 1.5.1 oder höher als Ziel haben (indem Sie den entsprechenden Namespace-Wert in der Anwendungsdeskriptordatei angeben).

In der folgenden vereinfachten Anwendung wird die reason-Eigenschaft von InvokeEvent verwendet, um das Verhalten für ein Aufrufereignis zu bestimmen. Wenn die reason-Eigenschaft den Wert "login" hat, bleibt die Anwendung im Hintergrund. Andernfalls wird die Hauptanwendung sichtbar gemacht. Eine Anwendung, die dieses Muster verwendet, wird normalerweise bei der Anmeldung gestartet, damit Hintergrundverarbeitung oder Ereignisüberwachung stattfinden kann, und öffnet ein Fenster, wenn der Benutzer die Anwendung aufgerufen hat.

```
package {
    import flash.desktop.InvokeEventReason;
    import flash.desktop.NativeApplication;
    import flash.display.Sprite;
    import flash.events.InvokeEvent;

    public class StartAtLogin extends Sprite
    {
        public function StartAtLogin()
        {
            try
            {
                NativeApplication.nativeApplication.startAtLogin = true;
            }
            catch ( e:Error )
            {
                trace( "Cannot set startAtLogin:" + e.message );
            }

            NativeApplication.nativeApplication.addEventListener( InvokeEvent.INVOKE, onInvoke );
        }

        private function onInvoke( event:InvokeEvent ):void
        {
            if( event.reason == InvokeEventReason.LOGIN )
            {
                //do background processing...
                trace( "Running in background..." );
            }
            else
            {
                this.stage.nativeWindow.activate();
            }
        }
    }
}
```

Hinweis: *Um das unterschiedliche Verhalten zu sehen, verpacken und installieren Sie die Anwendung. Die* `startAtLogin`*-Eigenschaft kann nur für installierte Anwendungen gesetzt werden.*

Aufrufen von AIR-Anwendungen aus dem Browser

Adobe AIR 1.0 und höher

Mit der Funktion für den Browseraufruf kann eine Website eine installierte AIR-Anwendung vom Browser aus aufrufen. Das Aufrufen vom Browser ist nur zulässig, wenn die Anwendungsdeskriptordatei `allowBrowserInvocation` auf `true` setzt:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Wird die Anwendung über den Browser aufgerufen, löst das NativeApplication-Objekt der Anwendung ein BrowserInvokeEvent-Objekt aus.

Rufen Sie die Methode `addEventListener()` des NativeApplication-Objekts (`NativeApplication.nativeApplication`) in der AIR-Anwendung auf, um BrowserInvokeEvent-Ereignisse anzunehmen. Wird ein Ereignis-Listener für ein BrowserInvokeEvent-Ereignis registriert, empfängt er auch alle BrowserInvokeEvent-Ereignisse, die vor der Registrierung aufgetreten sind. Diese Ereignisse werden nach der Rückgabe des Aufrufs von `addEventListener()` ausgelöst, jedoch nicht unbedingt vor anderen BrowserInvokeEvent-Ereignissen, die nach der Registrierung empfangen wurden. Auf diese Weise können Sie BrowserInvokeEvent-Ereignisse verarbeiten, die aufgetreten sind, bevor Ihr Initialisierungscode ausgeführt wird (zum Beispiel, wenn die Anwendung ursprünglich vom Browser aufgerufen wurde). Denken Sie daran, dass beim Hinzufügen eines Ereignis-Listeners zu einem späteren Zeitpunkt in der Ausführung (nach der Initialisierung der Anwendung) dieser immer noch alle BrowserInvokeEvent-Ereignisse empfängt, die seit dem Starten der Anwendung aufgetreten sind.

Das BrowserInvokeEvent-Objekt hat folgende Eigenschaften:

Eigenschaft	**Beschreibung**
arguments	Ein Array von Argumenten (Strings), die an die Anwendung zu übergeben sind.
isHTTPS	Ob der Browserinhalt das https-URL-Schema verwendet (`true`) oder nicht (`false`).
isUserEvent	Ob der Browseraufruf in einem Benutzerereignis resultiert (z. B. Mausklick). In AIR 1.0 ist dies immer auf `true` gesetzt; AIR benötigt ein Benutzerereignis für die Browseraufruffunktion.
sandboxType	Der Sandbox-Typ für den Browserinhalt. Gültige Werte werden so definiert wie die Werte, die in der Eigenschaft `Security.sandboxType` verwendet werden. Folgende Werte stehen zur Verfügung: • `Security.APPLICATION` – Die Inhalte befinden sich in der Sicherheits-Sandbox der Anwendung. • `Security.LOCAL_TRUSTED` – Die Inhalte befinden sich in der local-with-filesystem-Sicherheits-Sandbox. • `Security.LOCAL_WITH_FILE` – Die Inhalte befinden sich in der local-with-filesystem-Sicherheits-Sandbox. • `Security.LOCAL_WITH_NETWORK` – Die Inhalte befinden sich in der local-with-networking-Sicherheits-Sandbox. • `Security.REMOTE` – Die Inhalte befinden sich in einer Remote(-Netzwerk)-Domäne.
securityDomain	Die Sicherheitsdomäne für den Browserinhalt, zum Beispiel `www.adobe.com` oder `www.example.org`. Diese Eigenschaft wird nur für Inhalte in der Remote-Sicherheits-Sandbox gesetzt (für Inhalte aus einer Netzwerk-Domäne). Sie wird nicht für Inhalte in einer lokalen Sicherheits-Sandbox oder der Anwendungs-Sicherheits-Sandbox gesetzt.

Stellen Sie bei Verwendung der Browseraufrufsfunktion sicher, dass alle Sicherheitsüberlegungen beachtet wurden. Wenn eine Website eine AIR-Anwendung aufruft, kann sie über die Eigenschaft `arguments` des BrowserInvokeEvent-Objekts Daten senden. Seien Sie vorsichtig, wenn diese Daten bei sicherheitsrelevanten Vorgängen wie Dateien oder Code, die APIs laden, verwendet werden. Wie hoch das Risiko ist, hängt davon ab, was die Anwendung mit den Daten macht. Wenn Sie davon ausgehen, dass die Anwendung nur von einer bestimmten Website aufgerufen wird, sollte die Anwendung die Eigenschaft `securityDomain` des BrowserInvokeEvent-Objekts prüfen. Sie können auch zur Bedingung machen, dass die Website, die die Anwendung aufruft, HTTPs verwendet. Überprüfen Sie dies mit der Eigenschaft `isHTTPS` des BrowserInvokeEvent-Objekts.

Die Anwendung muss die übergebenen Daten prüfen. Geht die Anwendung zum Beispiel davon aus, dass URLs an eine spezifische Domäne übergeben werden, muss sie prüfen, dass die URLs tatsächlich auf diese Domäne verweisen. So kann vermieden werden, dass ein Angreifer die Anwendung dazu bringt, ihm sicherheitsrelevante Daten zu senden.

Anwendungen sollten keine BrowserInvokeEvent-Argumente verwenden, die auf lokale Ressourcen verweisen. So darf eine Anwendung keine File-Objekte erstellen, die auf einem Pfad basieren, der vom Browser übergeben wurde. Wird davon ausgegangen, das Remote-Pfadangaben vom Browser übergeben werden, muss die Anwendung sicherstellen, dass diese Pfadangaben nicht das Protokoll `file://` anstelle eines Remote-Protokolls verwenden.

Schließen der Anwendung

Adobe AIR 1.0 und höher

Eine Anwendung lässt sich am schnellsten mit einem Aufruf der exit()-Methode des NativeApplication-Objekts schließen. Dies funktioniert problemlos, wenn die Anwendung keine Daten speichern oder externe Ressourcen bereinigen muss. Durch den Aufruf von `exit()` werden alle Fenster geschlossen und die Anwendung wird beendet. Um jedoch den Fenstern und anderen Anwendungskomponenten die Möglichkeit zu geben, den Vorgang zu unterbrechen, z. B. um wichtige Daten zu speichern, lösen Sie vor dem Aufruf von `exit()` die entsprechenden Warnereignisse aus.

Es ist sinnvoll, das Verlassen einer Anwendung so zu gestalten, dass die Ausführung immer gleich abläuft, unabhängig davon, wie der Beendigungsvorgang gestartet wurde. Der Benutzer (oder das Betriebssystem) kann die Beendigung der Anwendung auf folgende Arten einleiten:

- Durch Schließen des letzten Anwendungsfensters, wenn `NativeApplication.nativeApplication.autoExit` auf `true` gesetzt ist.
- Durch Auswahl des Befehls „Schließen“ im Betriebssystem, wenn der Benutzer zum Beispiel den Befehl für das Schließen der Anwendung aus dem Standardmenü wählt. (Dies geschieht nur auf Mac-Betriebssystemen. Windows und Linux stellen keinen Befehl für das Schließen der Anwendung über die Benutzeroberfläche des Systems zur Verfügung.)
- Durch Herunterfahren des Computers.

Wenn der Befehl zum Schließen vom Betriebssystem über eines dieser Verfahren geleitet wird, löst NativeApplication ein `exiting`-Ereignis aus. Wird dieses `exiting`-Ereignis nicht von einem Listener unterbrochen, werden alle offenen Fenster geschlossen. Jedes Fenster löst ein `closing`- und dann ein `close`-Ereignis aus. Unterbricht eines der Fenster das `closing`-Ereignis, wird das Schließen der Anwendung gestoppt.

Wenn die Reihenfolge, in der die Fenster geschlossen werden, für die Anwendung wichtig ist, richten Sie einen Listener für das Ereignis `exiting` von NativeApplication ein und schließen Sie die Fenster eigenhändig in der richtigen Reihenfolge. Dies ist zum Beispiel erforderlich, wenn ein Dokumentfenster mit Werkzeugpaletten vorhanden ist. Es hat vielleicht unangenehme oder noch schlimmere Folgen, wenn das System die Paletten schließt, der Benutzer jedoch beschlossen hat, den Beendigungsvorgang zu unterbrechen, um Daten zu speichern. Unter Windows tritt das Ereignis `exiting` nur auf, nachdem das letzte Fenster geschlossen wurde (wenn die Eigenschaft `autoExit` des NativeApplication-Objekts auf `true` gesetzt wurde).

Stellen Sie bei allen Schließvorgängen, gleichgültig ob Sie über die Oberfläche des Betriebssystems, Menübefehle oder die Anwendungslogik ausgelöst wurden, sicher, dass der Vorgang gleich abläuft, indem Sie folgende Regeln beachten:

1. Lösen Sie ein `exiting`-Ereignis immer durch das NativeApplication-Objekt aus, bevor Sie im Anwendungscode `exit()` aufrufen, und prüfen Sie, dass andere Komponenten der Anwendung das Ereignis nicht abbrechen.

```
public function applicationExit():void {
    var exitingEvent:Event = new Event(Event.EXITING, false, true);
    NativeApplication.nativeApplication.dispatchEvent(exitingEvent);
    if (!exitingEvent.isDefaultPrevented()) {
        NativeApplication.nativeApplication.exit();
    }
}
```

2 Verwenden Sie einen Listener für das Ereignis `exiting` des `NativeApplication.nativeApplication`-Objekts und schließen Sie in der Prozedur alle Fenster (lösen Sie erst ein `closing`-Ereignis aus). Nehmen Sie die notwendigen Bereinigungen, zum Beispiel das Speichern von Anwendungsdaten oder Löschen temporärer Dateien vor, nachdem alle Fenster geschlossen wurden. Verwenden Sie während der Bereinigung nur synchrone Methoden, um sicherzustellen, dass diese abgeschlossen wurden, bevor die Anwendung beendet wird.

   Wenn es gleichgültig ist, in welcher Reihenfolge die Fenster geschlossen werden, können Sie mit einer Schleife durch das Array `NativeApplication.nativeApplication.openedWindows` vorgehen und die Fenster nacheinander schließen. Wenn die Reihenfolge *wichtig ist*, stellen Sie einen Mechanismus zur Verfügung, um die Fenster in der richtigen Reihenfolge zu schließen.

```
private function onExiting(exitingEvent:Event):void {
    var winClosingEvent:Event;
    for each (var win:NativeWindow in NativeApplication.nativeApplication.openedWindows) {
        winClosingEvent = new Event(Event.CLOSING,false,true);
        win.dispatchEvent(winClosingEvent);
        if (!winClosingEvent.isDefaultPrevented()) {
            win.close();
        } else {
            exitingEvent.preventDefault();
        }
    }

    if (!exitingEvent.isDefaultPrevented()) {
        //perform cleanup
    }
}
```

3 Die Fenster sollten ihre eigene Bereinigung selbst handhaben, indem sie auf ihre jeweiligen `closing`-Ereignisse warten.

4 Verwenden Sie nur einen `exiting`-Listener in Ihrer Anwendung, da die zuvor aufgerufenen Prozeduren nicht wissen können, welche auf sie folgenden Prozeduren das `exiting`-Ereignis unterbrechen. (Auf die Reihenfolge der Ausführung sollte man sich nicht verlassen.)

Kapitel 51: Arbeiten mit AIR-Laufzeit- und Betriebssysteminformationen

Adobe AIR 1.0 und höher

In diesem Kapitel werden Verfahren behandelt, mit denen eine AIR-Anwendung Dateiverknüpfungen des Betriebssystems verwalten, Benutzeraktivitäten erkennen und Informationen über die Adobe® AIR®-Laufzeitumgebung abrufen kann.

Verwandte Hilfethemen

flash.desktop.NativeApplication

Verwalten von Dateiverknüpfungen

Adobe AIR 1.0 und höher

Verknüpfungen zwischen der Anwendung und einem Dateityp müssen im Anwendungsdeskriptor deklariert werden. Während des Installationsvorgangs verknüpft das AIR-Installationsprogramm die AIR-Anwendung mit der Standardanwendung zum Öffnen aller deklarierten Dateitypen, sofern keine andere Anwendung bereits als Standardanwendung gilt. Der AIR-Installationsvorgang überschreibt keine bestehenden Dateitypverknüpfungen. Rufen Sie zur Laufzeit die `NativeApplication.setAsDefaultApplication()`-Methode auf, um die Verknüpfung aus einer anderen Anwendung zu übernehmen.

Es ist eine gute Angewohnheit zu prüfen, ob die erwarteten Dateiverknüpfungen beim Starten der Anwendung aktiv sind. Grund hierfür ist, dass das Installationsprogramm der AIR-Anwendung bestehende Dateiverknüpfungen nicht überschreibt und sich die Dateiverknüpfungen auf dem System eines Benutzers jederzeit ändern können. Wenn für eine andere Anwendung die aktuelle Dateiverknüpfung gilt, sollte der Benutzer schon aus Höflichkeit unterrichtet werden, bevor eine bestehende Verknüpfung übernommen wird.

Die folgenden Methoden der NativeApplication-Klasse ermöglichen einer Anwendung das Verwalten von Dateiverknüpfungen. Jede dieser Methoden verwendet die Dateierweiterung als Parameter:

Methode	Beschreibung
isSetAsDefaultApplication()	Gibt true zurück, wenn die AIR-Anwendung derzeit mit dem angegebenen Dateityp verknüpft ist.
setAsDefaultApplication()	Erstellt die Verknüpfung zwischen der AIR-Anwendung und der Aktion zum Öffnen des Dateityps.
removeAsDefaultApplication()	Entfernt die Verknüpfung zwischen der AIR-Anwendung und dem Dateityp.
getDefaultApplication()	Gibt den Pfad der Anwendung wieder, die derzeit mit dem Dateityp verknüpft ist.

AIR kann nur Verknüpfungen verwalten, die ursprünglich im Anwendungsdeskriptor deklariert wurden. Sie können auch dann keine Informationen über die Verknüpfungen eines nicht deklarierter Dateityps abrufen, wenn der Benutzer die Verknüpfung zwischen diesem Dateityp und der Anwendung manuell hergestellt hat. Beim Aufrufen einer der Methoden zum Verwalten von Dateiverknüpfungen für die Erweiterung eines nicht im Anwendungsdeskriptor deklarierten Dateityps wird eine Laufzeitausnahme ausgelöst.

Abrufen von Laufzeitversion und Patchebene

Adobe AIR 1.0 und höher

Das NativeApplication-Objekt verfügt über einer`runtimeVersion`-Eigenschaft, die der Version der Laufzeitumgebung entspricht, in der die Anwendung ausgeführt wird (ein String wie z. B. `"1.0.5"`). Das NativeApplication-Objekt verfügt ferner über eine `runtimePatchLevel`-Eigenschaft, die der Patchebene der Laufzeitumgebung entspricht (eine Zahl wie z. B. 2960). Der folgende Code verwendet diese Eigenschaften:

```
trace(NativeApplication.nativeApplication.runtimeVersion);
trace(NativeApplication.nativeApplication.runtimePatchLevel);
```

Erkennen von AIR-Funktionalität

Adobe AIR 1.0 und höher

Für eine mit der Adobe AIR-Anwendung gebundelten Datei ist für die `Security.sandboxType`-Eigenschaft der von der `Security.APPLICATION`-Konstante definierte Wert festgelegt. Sie können Inhalt (der AIR-spezifische APIs enthalten kann, aber nicht enthalten muss) wie im nachstehenden Code veranschaulicht basierend darauf laden, ob eine Datei in der Adobe AIR-Sicherheits-Sandbox enthalten ist:

```
if (Security.sandboxType == Security.APPLICATION)
{
    // Load SWF that contains AIR APIs
}
else
{
    // Load SWF that does not contain AIR APIs
}
```

Alle nicht mit der AIR-Anwendung installierten Ressourcen werden den gleichen Sicherheits-Sandboxen zugewiesen, die auch von Adobe® Flash® Player in einem Webbrowser zugewiesen würden. Remoteressourcen werden Sandboxen entsprechend der Quelldomäne und lokale Ressourcen der local-with-networking-, local-with-filesystem- oder local-trusted-Sandbox zugewiesen.

Sie können prüfen, ob für die statische `Capabilities.playerType`-Eigenschaft `"Desktop"` festgelegt wurde, um herauszufinden, ob Inhalt in der Laufzeitumgebung (und nicht in dem im Browser ausgeführten Flash Player) ausgeführt wird.

Weitere Informationen finden Sie unter „AIR-Sicherheit“ auf Seite 1153.

Nachverfolgen der Benutzerpräsenz

Adobe AIR 1.0 und höher

Das NativeApplication-Objekt löst zwei Ereignisse aus, mit denen Sie erkennen können, ob ein Benutzer einen Computer aktiv nutzt. Wird während des von der `NativeApplication.idleThreshold`-Eigenschaft festgelegten Intervalls keine Maus- oder Tastaturaktivität erkannt, löst das NativeApplication-Objekt ein `userIdle`-Ereignis aus. Bei der nächsten Eingabe über die Tastatur oder Maus löst das NativeApplication-Objekt ein `userPresent`-Ereignis aus. Das `idleThreshold`-Intervall wird in Sekunden gemessen und hat einen Standardwert von 300 (5 Minuten). Mit der `NativeApplication.nativeApplication.lastUserInput`-Eigenschaft können Sie auch die seit der letzten Benutzereingabe verstrichenen Sekunden abrufen.

Die folgenden Codezeilen legen eine Leerlaufzeit von 2 Minuten fest und überwachen sowohl das `userIdle`- als auch das`userPresent` -Ereignis:

```
NativeApplication.nativeApplication.idleThreshold = 120;
NativeApplication.nativeApplication.addEventListener(Event.USER_IDLE, function(event:Event) {
    trace("Idle");
});
NativeApplication.nativeApplication.addEventListener(Event.USER_PRESENT,
function(event:Event) {
    trace("Present");
});
```

Hinweis: *Zwischen zwei beliebigen `userPresent`-Ereignissen wird nur ein `userIdle` ausgelöst.*

Kapitel 52: Arbeiten mit nativen AIR-Fenstern

Adobe AIR 1.0 und höher

Mithilfe der Klassen, die von der Adobe® AIR® nativen Fenster-API bereitgestellt werden, können Sie Desktopfenster erstellen und verwalten.

Grundlagen von nativen Fenstern in AIR

Adobe AIR 1.0 und höher

Kurzbeschreibungen und Codebeispiele für die Arbeit mit nativen Fenstern in AIR finden Sie in den folgenden Kurzanleitungen in der Adobe Developer Connection:

- Erstellen einer transparenten Fensteranwendung (Flex)
- Interagieren mit Fenstern (Flex)
- Anpassen des Gesamterscheinungsbilds von nativen Fenstern (Flex)
- Starten von Fenstern (Flex)
- Erstellen überlappender Fenster (Flex)
- Steuern der Anzeigereihenfolge von Fenstern (Flex)
- Erstellen von nicht-rechteckigen, skalierbaren Fenstern (Flex)
- Interagieren mit Fenstern (Flash)
- Anpassen des Gesamterscheinungsbilds von nativen Fenstern (Flash)
- Erstellen überlappender Fenster (Flash)
- Steuern der Anzeigereihenfolge von Fenstern (Flash)
- Erstellen von nicht-rechteckigen, skalierbaren Fenstern (Flash)

AIR bietet eine benutzerfreundliche, plattformübergreifende Fenster-API, die mithilfe von Flash®-, Flex™- und HTML-Programmiertechniken das Erstellen von nativen Betriebssystemfenstern ermöglicht.

AIR gewährt Ihnen viel Spielraum bei der Entwicklung des Erscheinungsbilds einer Anwendung. Die von Ihnen erstellten Fenster können wie bei einer standardmäßigen Desktopanwendung aussehen, bei der Ausführung auf einem Mac dem Apple-Stil und bei der Ausführung unter Windows den Microsoft-Konventionen bzw. unter Linux dem Fenstermanager entsprechen, ohne dass Sie eine einzige Zeile mit plattformspezifischen Code einfügen müssen. Oder entwickeln Sie Ihren eigenen Stil unabhängig von der Plattform mithilfe des vom Flex-Framework gebotenen erweiterbaren Fensterdesigns, dem Sie Skins zuweisen können. Sie können sogar Ihre eigenen Fensterdesigns mit Vektor- und Bitmapgrafiken zeichnen, und zwar mit voller Unterstützung für Transparenz und Alpha-Mischung in Bezug zum Desktop. Haben Sie genug von rechteckigen Fenstern? Zeichnen Sie ein rundes.

Fenster in AIR

Adobe AIR 1.0 und höher

AIR unterstützt drei spezielle APIs zur Arbeit mit Fenstern:

- Die an ActionScript ausgerichtete NativeWindow-Klasse bietet die Fenster-API auf der untersten Ebene. Verwenden Sie NativeWindows in Anwendungen, die mit ActionScript und Flash Professional erstellt werden. Sie können die NativeWindow-Klasse erweitern, um die in Ihrer Anwendung verwendeten Fenster zu spezialisieren.
- In der HTML-Umgebung können Sie die JavaScript Window-Klasse verwenden, wie Sie dies in einer browserbasierten Webanwendung tun würden. Aufrufe von JavaScript Window-Methoden werden an das zugrundeliegende native Fensterobjekt weitergeleitet.
- Das Flex-Framework mx:WindowedApplication und mx:Window-Klassen bieten einen Flex-„Wrapper“ für die NativeWindow-Klasse. Die WindowedApplication-Komponente ersetzt die Application-Komponenten, wenn Sie eine AIR-Anwendung mit Flex erstellen. Sie muss immer als Startfenster in Ihrer Flex-Anwendung verwendet werden.

ActionScript-Fenster

Verwenden Sie bei der Erstellung von Fenstern mit der NativeWindow-Klasse direkt die Bühne und Anzeigeliste in Flash Player. Um einem NativeWindow ein visuelles Objekt hinzuzufügen, fügen Sie das Objekt zur Anzeigeliste der Fensterbühne oder zu einem anderen Anzeigeobjektcontainer auf der Bühne hinzu.

HTML-Fenster

Beim Erstellen von HTML-Fenstern wird Inhalt mithilfe von HTML, CSS und JavaScript angezeigt. Um einem HTML-Fenster ein visuelles Objekt hinzuzufügen, fügen Sie den Inhalt zum HTML DOM hinzu. HTML-Fenster sind eine spezielle Kategorie von „NativeWindow“. Der AIR-Host definiert eine `nativeWindow`-Eigenschaft in den HTML-Fenstern, die den Zugriff auf die zugrunde liegende NativeWindow-Instanz ermöglicht. Mithilfe dieser Eigenschaft können Sie auf die hier beschriebenen NativeWindow-Eigenschaften, -Methoden und -Ereignisse zugreifen.

Hinweis: *Das Window-Objekt von JavaScript bietet ebenfalls Methoden zur Skripterstellung der enthaltenden Fenster, wie z. B.* `moveTo()` *und* `close()`*. Wenn überlappende Methoden verfügbar sind, können Sie die praktischere auswählen.*

Flex Framework-Fenster

Beim Erstellen von Fenstern mit dem Flex-Framework werden die Fenster in der Regel mit MXML-Komponenten gefüllt. Um einem Fenster eine Flex-Komponente hinzuzufügen, fügen Sie das Komponentenelement der MXML-Definition des Fensters zu. Außerdem können Sie Inhalt dynamisch mit ActionScript hinzufügen. Die mx:WindowedApplication- und mx:Window-Komponenten dienen als Flex-Behälter. Daher können sie im Gegensatz zu NativeWindow-Objekten Flex-Komponenten direkt akzeptieren. Bei Bedarf können Sie auf die NativeWindow-Eigenschaften und -Methoden über die WindowedApplication- und Window-Objekte mithilfe der `nativeWindow`-Eigenschaft zugreifen.

Das ursprüngliche Anwendungsfenster

Das erste Fenster der Anwendung wird automatisch von AIR erstellt. AIR legt die Eigenschaften und den Inhalt des Fensters anhand der im `initialWindow`-Element der Anwendungsdeskriptordatei angegebenen Parameter fest.

Wenn es sich beim Stamminhalt um eine SWF-Datei handelt, erstellt AIR eine NativeWindow-Instanz, lädt die SWF-Datei und fügt sie zur Fensterbühne hinzu. Wenn es sich beim Stamminhalt um eine HTML-Datei handelt, erstellt AIR ein HTML-Fenster und lädt die HTML-Datei.

Native Fenster-Klassen

Adobe AIR 1.0 und höher

Die native Fenster-API enthält die folgenden Klassen:

Paket	Klassen
flash.display	• NativeWindow • NativeWindowInitOptions • NativeWindowDisplayState • NativeWindowResize • NativeWindowSystemChrome • NativeWindowType
flash.events	• NativeWindowBoundsEvent • NativeWindowDisplayStateEvent

Ereignisfluss bei nativen Fenstern

Adobe AIR 1.0 und höher

Native Fenster lösen Ereignisse aus, um interessierte Komponenten über bevorstehende oder bereits eingetretene wichtige Änderungen zu informieren. Viele auf Fenster bezogene Ereignisse werden paarweise ausgelöst. Mit dem ersten Ereignis wird vor einer bevorstehenden Änderung gewarnt. Das zweite Ereignis teilt mit, dass die Änderung vorgenommen wurde. Ein Warnereignis kann abgebrochen werden, ein Benachrichtigungsereignis jedoch nicht. Im Folgenden wird der Ereignisablauf dargestellt, der auftritt, wenn ein Benutzer auf die Schaltfläche zum Maximieren eines Fensters klickt:

1. Das NativeWindow-Objekt löst ein `displayStateChanging`-Ereignis aus.
2. Wenn das Ereignis nicht von einem registrierten Listener abgebrochen wird, wird das Fenster maximiert.
3. Das NativeWindow-Objekt löst ein `displayStateChange`-Ereignis aus.

   Außerdem löst das NativeWindow-Objekt Ereignisse zu damit im Zusammenhang stehenden Änderungen an der Fenstergröße und -position aus. Zu diesen im Zusammenhang stehenden Änderungen löst das Fenster keine Warnereignisse aus. Die im Zusammenhang stehenden Ereignisse sind:

   a. Ein `move`-Ereignis wird ausgelöst, wenn die obere linke Ecke des Fensters aufgrund der Maximierung verschoben wird.
   b. Ein `resize`-Ereignis wird ausgelöst, wenn sich die Fenstergröße aufgrund der Maximierung geändert hat.

   Ein NativeWindow-Objekt löst eine ähnliche Abfolge von Ereignissen aus, wenn ein Fenster minimiert, wiederhergestellt, geschlossen, verschoben und skaliert wird.

   Die Warnereignisse werden nur ausgelöst, wenn eine Änderung durch das Fensterdesign oder einen anderen vom Betriebssystem gesteuerten Mechanismus eingeleitet wird. Wenn Sie eine Fenstermethode aufrufen, um die Größe, die Position oder den Anzeigestatus des Fensters zu ändern, wird nur ein Ereignis ausgelöst, um die Änderung anzukündigen. Auf Wunsch können Sie mithilfe der `dispatchEvent()`-Fenstermethode ein Warnereignis auslösen. Anschließend können Sie prüfen, ob das Warnereignis abgebrochen wurde, bevor die Änderung ausgeführt wird.

Ausführliche Informationen über die Klassen, Methoden, Eigenschaften und Ereignisse der Fenster-API finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Eigenschaften zur Steuerung von Stil und Verhalten nativer Fenster

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mithilfe der folgenden Eigenschaften wird das grundlegende Erscheinungsbild und Verhalten eines Fensters gesteuert:

- `type`
- `systemChrome`
- `transparent`
- `owner`

Diese Eigenschaften legen Sie beim Erstellen eines Fensters im NativeWindowInitOptions-Objekt fest, das an den Fensterkonstruktor übergeben wird. AIR entnimmt der Anwendungsdeskriptordatei die Eigenschaften für das ursprüngliche Anwendungsfenster. (Eine Ausnahme bildet die `type`-Eigenschaft, die in der Anwendungsdeskriptordatei nicht eingestellt werden kann und immer auf `normal` eingestellt ist.) Die Eigenschaften können nach der Fenstererstellung nicht mehr geändert werden.

Einige Einstellungen dieser Eigenschaften schließlich sich gegenseitig aus: Sie können `systemChrome` nicht auf `standard` einstellen, wenn `transparent` den Wert `true` oder `type` den Wert `lightweight` aufweist.

Fenstertypen

Adobe AIR 1.0 und höher

In den AIR-Fenstertypen werden Fensterdesign- und Sichtbarkeitsattribute des nativen Betriebssystems kombiniert, um drei Fensterfunktionstypen zu erstellen. Verwenden Sie die in der NativeWindowType-Klasse definierten Konstanten, um die Typennamen im Code anzugeben. AIR stellt die folgenden Fenstertypen bereit:

Typ	Beschreibung
Normal	Ein typisches Fenster. Normale Fenster verwenden das Fensterdesign in voller Größe und werden in der Windows-Taskleiste und im Fenster-Menü von Mac OS X angezeigt.
Utility	Eine Werkzeugpalette. Dienstprogrammfenster verwenden eine schmalere Version des System-Fensterdesigns und werden nicht in der Windows-Taskleiste und im Fenster-Menü von Mac OS X angezeigt.
Lightweight	Lightweight-Fenster weisen kein Fensterdesign auf und werden nicht in der Windows-Taskleiste und im Fenster-Menü von Mac OS X angezeigt. Zudem verfügen Lightweight-Fenster unter Windows nicht über das System-Menü (Alt+Leertaste). Lightweight-Fenster eignen sich für Benachrichtigungen und Steuerungen wie Kombinationsfelder, die einen Anzeigebereich öffnen, der nur für einen kurzen Zeitraum eingeblendet wird. Bei Verwendung des `lightweight`-Typs muss `systemChrome` auf `none` eingestellt werden.

Fensterdesign

Adobe AIR 1.0 und höher

Beim Fensterdesign handelt es sich um die Steuerelemente, die es Benutzern ermöglichen, ein Fenster in der Desktopumgebung zu bearbeiten. Zu den Fensterdesignelementen gehören die Titelleiste, die Schaltflächen der Titelleiste, der Rahmen und Haltegriffe zur Größenänderung.

System-Fensterdesign

Sie können die `systemChrome`-Eigenschaft auf `standard` oder `none` einstellen. Wählen Sie `standard`, wenn Sie dem Fenster die vom Betriebssystem des Benutzers stammenden Standardsteuerelemente zuweisen möchten. Wählen Sie `none`, wenn Sie dem Fenster Ihre eigenen Steuerelemente zuweisen möchten. Verwenden Sie die in der NativeWindowSystemChrome-Klasse definierten Konstanten, um die Einstellungen für das System-Fensterdesign im Code anzugeben.

Das System-Fensterdesign wird vom System verwaltet. Ihre Anwendung hat keinen direkten Zugriff auf die Steuerelemente selbst, kann aber auf die Ereignisse reagieren, die bei der Verwendung der Steuerelemente ausgelöst werden. Wenn Sie einem Fenster das Standard-Fensterdesign zuweisen, müssen Sie die `transparent`-Eigenschaft auf den Wert `false` und die `type`-Eigenschaft auf den Wert `normal` oder `utility` einstellen.

Flex-Fensterdesign

Bei Verwendung der Flex-Komponenten „WindowedApplication“ oder „Window“ können Sie dem Fenster entweder das System-Fensterdesign oder das Fensterdesign des Flex-Frameworks zuweisen. Um das Flex-Fensterdesign zuzuweisen, müssen Sie die bei der Fenstererstellung verwendete `systemChrome`-Eigenschaft auf `none` einstellen. Bei Verwendung der Spark-Komponenten von Flex 4 anstelle der mx-Komponenten müssen Sie die Skin-Klasse angeben, damit das Flex-Fensterdesign verwendet werden kann. Sie können die integrierten oder eigene Skins verwenden. Im folgenden Beispiel wird gezeigt, wie die integrierte Spark-Skin-Klasse „WindowedApplication“ verwendet wird, um das Fensterdesign anzugeben:

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:s="library://ns.adobe.com/flex/spark"
xmlns:mx="library://ns.adobe.com/flex/mx">
<fx:Style>
@namespace "library://ns.adobe.com/flex/spark";
WindowedApplication
{
    skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
}
</fx:Style>
</s:WindowedApplication>
```

Weitere Informationen finden Sie in der folgenden Dokumentation: Using Flex 4: About the AIR window containers: Controlling window chrome.

Benutzerdefiniertes Fensterdesign

Wenn Sie ein Fenster erstellen und ihm nicht das System-Fensterdesign zuweisen, dann müssen Sie Ihr eigenes Fensterdesign hinzufügen, um die Interaktionen zwischen Benutzer und Fenster verarbeiten zu können. Sie haben außerdem die Möglichkeit, transparente, nicht rechteckige Fenster zu erstellen.

Um ein benutzerdefiniertes Fensterdesign mit der mx:WindowedApplication- oder mx:Window-Komponente zu verwenden, müssen Sie den `showFlexChrome`-Stil auf `false` einstellen. Andernfalls fügt Flex Ihren Fenstern sein eigenes Fensterdesign hinzu.

Fenstertransparenz

Adobe AIR 1.0 und höher

Um eine Alpha-Mischung eines Fensters mit dem Desktop oder anderen Fenstern zu ermöglichen, müssen Sie die `transparent`-Eigenschaft des Fensters auf `true` setzen. Die `transparent`-Eigenschaft muss vor dem Erstellen des Fensters eingestellt werden und kann nicht geändert werden.

Ein transparentes Fenster weist keinen Standardhintergrund auf. Jeder Fensterbereich, der kein von der Anwendung gezeichnetes Objekt enthält, ist unsichtbar. Weist ein angezeigtes Objekt eine Alpha-Einstellung von weniger als Eins auf, ist alles, was sich darunter befindet, sichtbar, auch andere Anzeigeobjekte im selben Fenster, andere Fenster und der Desktop.

Transparente Fenster eignen sich, wenn Sie Anwendungen erstellen möchten, deren Rahmen unregelmäßig geformt sind, die „ausgeblendet" werden oder unsichtbar zu sein scheinen. Die Wiedergabe großer Bereiche mit Alpha-Mischung kann jedoch viel Zeit beanspruchen, daher empfiehlt sich ein sparsamer Einsatz dieses Effekts.

Wichtig: *Unter Linux werden Mausereignisse nicht durch vollständig transparente Pixel weitergegeben. Sie sollten keine Fenster mit großen, vollständig transparenten Bereichen erstellen, da so möglicherweise der Benutzerzugriff auf andere Fenster oder Elemente auf dem Desktop verhindert wird. Unter Mac OS X und Windows werden Mausereignisse durch vollständig transparente Pixel weitergegeben.*

Transparenz kann nicht mit Fenstern verwendet werden, denen das System-Fensterdesign zugewiesen wurde. Außerdem wird SWF- und PDF-Inhalt in HTML möglicherweise nicht in transparenten Fenstern angezeigt. Weitere Informationen finden Sie unter „Erwägungen beim Laden von SWF- oder PDF-Inhalt in eine HTML-Seite" auf Seite 1074.

Die statische `NativeWindow.supportsTransparency`-Eigenschaft meldet, ob Fenstertransparenz verfügbar ist. Wenn Transparenz nicht unterstützt wird, wird die Anwendung vor einem schwarzen Hintergrund zusammengesetzt. In solchen Fällen werden transparente Bereiche der Anwendung in opakem Schwarz angezeigt. Für den Fall, dass diese Eigenschaft mit `false` getestet wird, empfiehlt es sich, für eine Ausweichlösung zu sorgen. Sie könnten beispielsweise eine Warnung für den Benutzer oder eine rechteckige, nicht transparente Benutzeroberfläche anzeigen.

Beachten Sie, dass Transparenz von den Betriebssystemen Mac und Windows immer unterstützt wird. Für die Unterstützung von Linux-Betriebssystemen ist ein Compositing-Fenstermanager erforderlich, aber selbst mit aktivem Compositing-Fenstermanager ist Transparenz aufgrund von Benutzeranzeigeoptionen oder der Hardwarekonfiguration möglicherweise nicht verfügbar.

Transparenz in einem MXML-Anwendungsfenster

Adobe AIR 1.0 und höher

Der Hintergrund eines MXML-Fensters ist in der Standardeinstellung opak, selbst wenn es als *transparent* erstellt wurde. (Beachten Sie den Transparenzeffekt an den Ecken des Fensters.) Um das Fenster mit einem transparenten Hintergrund anzuzeigen, müssen Sie eine Hintergrundfarbe und einen Alpha-Wert im Stylesheet oder in dem in der MXML-Anwendungsdatei enthaltenen <mx:Style>-Element einstellen. Mit dem folgenden Stylesheet erhält der Hintergrund eine leicht transparente grüne Farbe:

```
WindowedApplication
{
    background-alpha:".8";
    background-color:"0x448234";
}
```

Transparenz in einem HTML-Anwendungsfenster

Adobe AIR 1.0 und höher

In der Standardeinstellung wird bei HTML-Inhalt, der in HTML-Fenstern und HTMLLoader-Objekten angezeigt wird, ein opaker Hintergrund angezeigt, selbst wenn das enthaltende Fenster transparent ist. Um diesen standardmäßig bei HTML-Inhalt angezeigten Hintergrund zu deaktivieren, müssen Sie die `paintsDefaultBackground`-Eigenschaft auf `false` einstellen. Im folgenden Beispiel wird ein „HTMLLoader" erstellt und der Standardhintergrund wird deaktiviert:

```
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.paintsDefaultBackground = false;
```

In diesem Beispiel wird der Standardhintergrund eines HTML-Fensters mithilfe von JavaScript deaktiviert:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Wenn durch ein Element im HTML-Dokument eine Hintergrundfarbe eingestellt wird, ist der Hintergrund des Elements nicht transparent. Die Einstellung eines teilweisen Transparenzwerts (oder Deckkraftwerts) wird nicht unterstützt. Indem Sie eine transparente Grafik im PNG-Format als Hintergrund für eine Seite oder ein Seitenelement festlegen, können Sie einen ähnlichen optischen Effekt erzielen.

Fensterbesitz

Ein Fenster kann ein oder mehrere andere Fenster *besitzen*. Fenster, die sich im Besitz eines anderen Fensters befinden, werden immer vor dem übergeordneten Fenster angezeigt. Außerdem werden sie mit dem übergeordneten Fenster minimiert, wiederhergestellt und geschlossen. Der Besitz an einem Fenster kann nicht an ein anderes Fenster übertragen oder aufgehoben werden. Ein Fenster kann sich nur im Besitz eines übergeordneten Fensters befinden; es kann jedoch eine beliebige Anzahl anderer Fenster besitzen.

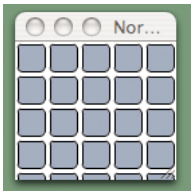
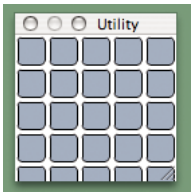
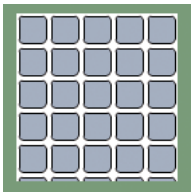
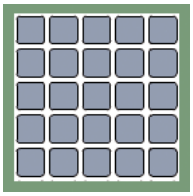
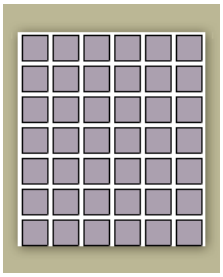
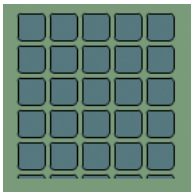
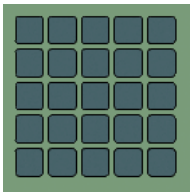
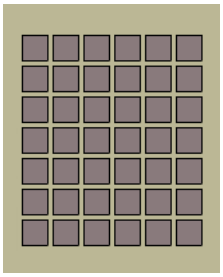
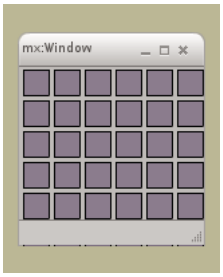
Mithilfe des Fensterbesitzes lassen sich Fenster, die für Werkzeugpaletten und Dialogfelder verwendet werden, einfacher verwalten. Wenn Sie beispielsweise ein Dialogfeld „Speichern“ zusammen mit einem Dokumentfenster anzeigen möchten, können Sie festlegen, dass das Dokumentfenster das Dialogfeld besitzt, damit das Dialogfeld immer automatisch vor dem Dokumentfenster angezeigt wird.

- NativeWindow.owner
- Christian Cantrell: Owned windows in AIR 2.6

Übersicht über Fenstereinstellungen

Adobe AIR 1.0 und höher

Die folgende Tabelle zeigt die optischen Effekte, die mit den verschiedenen Kombinationen von Eigenschaftseinstellungen für Fenster unter den Betriebssystemen Mac OS X, Windows und Linux erzielt werden:

Fenstereinstellungen	Mac OS X	Microsoft Windows	Linux*
Type: normal SystemChrome: standard Transparent: false			
Type: utility SystemChrome: standard Transparent: false			
Type: Beliebig SystemChrome: none Transparent: false			
Type: Beliebig SystemChrome: none Transparent: true			
mxWindowedApplication oder mx:Window Type: Beliebig SystemChrome: none Transparent: true			

Ubuntu mit Compiz-Fenstermanager

Hinweis: *Die folgenden Elemente des System-Fensterdesigns werden von AIR nicht unterstützt: die Mac OS X-Symbolleiste, das Mac OS X-Proxy-Symbol, Windows-Titelleistensymbole und alternative System-Fensterdesigns.*

Erstellen von Fenstern

Adobe AIR 1.0 und höher

AIR erstellt automatisch das erste Fenster einer Anwendung, bei Bedarf können Sie jedoch weitere Fenster erstellen. Mithilfe der NativeWindow-Konstruktormethode können Sie ein natives Fenster erstellen.

Sie können ein HTML-Fenster entweder mithilfe der `createRootWindow()`-Methode von „HTMLLoader“ erstellen oder indem Sie von einem HTML-Dokument aus die `window.open()`-Methode von JavaScript aufrufen. Das erstellte Fenster ist ein NativeWindow-Objekt, dessen Anzeigeliste ein HTMLLoader-Objekt enthält. Das HTMLLoader-Objekt interpretiert den HTML- und JavaScript-Inhalt für das Fenster und zeigt ihn an. Über die `window.nativeWindow`-Eigenschaft haben Sie in JavaScript Zugriff auf die Eigenschaften des zugrundeliegenden NativeWindow-Objekts. (Auf diese Eigenschaft kann nur Code, der in der AIR-Anwendungs-Sandbox ausgeführt wird, zugreifen.)

Bei der Initialisierung eines Fensters (auch des ersten Anwendungsfensters) kann die folgende Vorgehensweise empfehlenswert sein: Erstellen Sie das Fenster im unsichtbaren Status, laden Sie den Inhalt oder führen Sie Grafikaktualisierungen durch und machen Sie das Fenster erst dann sichtbar. Durch diese Vorgehensweise wird vermieden, dass Benutzer visuelle Verzerrungen auf dem Bildschirm sehen. Um festzulegen, dass das erste Anwendungsfenster im unsichtbaren Status erstellt wird, geben Sie das `<visible>false</visible>`-Tag im Anwendungsdeskriptor an. (Sie können das Tag auch ganz weglassen, da der Standardwert `false` lautet.) Neue NativeWindows-Instanzen sind standardmäßig unsichtbar. Wenn Sie ein HTML-Fenster mit der HTMLLoader-Methode `createRootWindow()` erstellen, können Sie das `visible`-Argument auf `false` setzen. Um ein Fenster sichtbar zu machen, rufen Sie die NativeWindow-Methode `activate()` auf oder stellen Sie die `visible`-Eigenschaft auf `true` ein.

Festlegen von Initialisierungseigenschaften für Fenster

Adobe AIR 1.0 und höher

Die Initialisierungseigenschaften eines nativen Fensters können nicht mehr geändert werden, nachdem das Desktopfenster erstellt wurde. Zu diesen unveränderlichen Eigenschaften und ihren Standardwerten gehören:

Eigenschaft	Standardwert
systemChrome	standard
type	normal
transparent	false
owner	null
maximizable	true
minimizable	true
resizable	true

Legen Sie die Eigenschaften für das erste von AIR erstellte Fenster in der Anwendungsdeskriptordatei fest. Das Hauptfenster einer AIR-Anwendung weist für die type-Eigenschaft immer den Wert *normal* auf. (In der Anwendungsdeskriptordatei können Sie weitere Fenstereigenschaften, wie etwa `visible`, `width` und `height` angeben, aber diese Eigenschaften können jederzeit geändert werden.)

Mithilfe der NativeWindowInitOptions-Klasse können Sie Eigenschaften für andere native und HTML-Fenster, die mithilfe Ihrer Anwendung erstellt wurden, festlegen. Beim Erstellen eines Fensters müssen Sie ein NativeWindowInitOptions-Objekt, das die Fenstereigenschaften angibt, entweder an die NativeWindow-Konstruktorfunktion oder die `createRootWindow()`-Methode von „HTMLLoader“ übergeben .

Im folgenden Beispiel wird ein NativeWindowInitOptions-Objekt für ein Dienstprogrammfenster erstellt:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.UTILITY
options.transparent = false;
options.resizable = false;
options.maximizable = false;
```

Die Einstellung von *systemChrome* auf *standard*, wenn *transparent* den Wert `true` oder `type` den Wert `lightweight` aufweist, *wird nicht unterstützt*.

Hinweis: *Für ein Fenster, das mithilfe der `window.open()`-Funktion von JavaScript erstellt wurde, können Sie keine Initialisierungseigenschaften festlegen. Sie können jedoch außer Kraft setzen, wie diese Fenster erstellt werden, indem Sie Ihre eigene HTMLHost-Klasse implementieren. Weitere Informationen finden Sie unter „Verarbeiten von JavaScript-Aufrufen von „window.open()““ auf Seite 1086.*

Bei dem Erstellen eines Fensters mithilfe der mx:Window-Klasse von Flex müssen Sie die Initialisierungseigenschaften im Fensterobjekt selbst angeben, und zwar entweder in der MXML-Deklaration für das Fenster oder im Code, mit dem das Fenster erstellt wird. Das zugrundeliegende NativeWindow-Objekt wird nicht erstellt, bis Sie die `open()`-Methode aufrufen. Nachdem ein Fenster geöffnet wurde, können die Initialisierungseigenschaften nicht mehr geändert werden.

Erstellen des ursprünglichen Anwendungsfensters

Adobe AIR 1.0 und höher

AIR erstellt das ursprüngliche Anwendungsfenster anhand der in der Anwendungsdeskriptordatei angegebenen Eigenschaften und lädt die Datei, auf die im content-Element verwiesen wird. Das Inhaltselement muss auf eine SWF- oder HTML-Datei verweisen.

Beim ursprünglichen Fenster kann es sich um das Hauptfenster der Anwendung handeln, oder es kann lediglich dazu dienen, eines oder mehrere andere Fenster zu starten. Es muss nicht unbedingt sichtbar sein.

Erstellen des ursprünglichen Fensters mit ActionScript

Adobe AIR 1.0 und höher

Beim Erstellen einer AIR-Anwendung mithilfe von ActionScript muss die Hauptklasse der Anwendung die Sprite-Klasse (oder eine Unterklasse der Sprite-Klasse) erweitern. Diese Klasse dient als Hauptzugangspunkt für die Anwendung.

Beim Starten der Anwendung erstellt AIR ein Fenster und eine Instanz der Hauptklasse und fügt diese Instanz der Fensterbühne hinzu. Um auf das Fenster zuzugreifen, warten Sie auf das `addedToStage`-Ereignis und rufen anschließend mit der `nativeWindow`-Eigenschaft des Stage-Objekts einen Verweis auf das NativeWindow-Objekt ab.

Im folgenden Beispiel wird die grundlegende Struktur der Hauptklasse einer mithilfe von ActionScript erstellten AIR-Anwendung verdeutlicht:

```
package {
    import flash.display.NativeWindow;
    import flash.display.Sprite;
    import flash.events.Event;

    public class MainClass extends Sprite
    {
        private var mainWindow:NativeWindow;
        public function MainClass(){
            this.addEventListener(Event.ADDED_TO_STAGE, initialize);
        }

        private function initialize(event:Event):void{
            mainWindow = this.stage.nativeWindow;
            //perform initialization...
            mainWindow.activate(); //show the window
        }
    }
}
```

Hinweis: *Technisch gesehen ist es möglich, auf die `nativeWindow`-Eigenschaft in der Konstruktorfunktion der Hauptklasse zuzugreifen. Dies ist jedoch ein besonderer Fall, der nur für das anfängliche Anwendungsfenster gilt.*

Wenn Sie eine Anwendung in Flash Professional erstellen, wird die Hauptdokumentklasse automatisch erstellt, wenn Sie keine eigene in einer separaten ActionScript-Datei erstellen. Über die `nativeWindow`-Bühneneigenschaft haben Sie Zugriff auf das NativeWindow-Objekt für das anfängliche Fenster. Mit dem folgenden Code wird zum Beispiel das Hauptfenster im maximierten Zustand aufgerufen (von der Zeitleiste).

```
import flash.display.NativeWindow;

var mainWindow:NativeWindow = this.stage.nativeWindow;
mainWindow.maximize();
mainWindow.activate();
```

Erstellen des ursprünglichen Fensters mit Flex

Adobe AIR 1.0 und höher

Beim Erstellen einer AIR-Anwendung mit dem Flex-Framework verwenden Sie die mx:WindowedApplication-Komponente als Stammelement Ihrer MXML-Hauptdatei. (Sie können auch die mx:Application-Komponente verwenden, sie unterstützt jedoch nicht alle in AIR verfügbaren Funktionen.) Die WindowedApplication-Komponente dient als erster Zugangspunkt für die Anwendung.

Beim Starten der Anwendung erstellt AIR ein natives Fenster, initialisiert das Flex-Framework und fügt der Fensterbühne das WindowedApplication-Objekt hinzu. Im Anschluss an die Startsequenz löst „WindowedApplication“ ein `applicationComplete`-Ereignis aus. Mit der `nativeWindow`-Eigenschaft der WindowedApplication-Instanz können Sie auf das Desktopfensterobjekt zugreifen.

Im folgenden Beispiel wird eine einfache WindowedApplication-Komponente erstellt, mit der die x- und y-Koordinaten eingestellt werden:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    applicationComplete="placeWindow()">
    <mx:Script>
        <![CDATA[
            private function placeWindow():void{
                this.nativeWindow.x = 300;
                this.nativeWindow.y = 300;
            }
        ]]>
    </mx:Script>
    <mx:Label text="Hello World" horizontalCenter="0" verticalCenter="0"/>
</mx:WindowedApplication>
```

Erstellen eines „NativeWindow“

Adobe AIR 1.0 und höher

Um ein „NativeWindow“ zu erstellen, übergeben Sie ein NativeWindowInitOptions-Objekt an den NativeWindow-Konstruktor:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.transparent = false;
var newWindow:NativeWindow = new NativeWindow(options);
```

Das Fenster wird erst angezeigt, wenn Sie die `visible`-Eigenschaft auf `true` einstellen oder die `activate()`-Methode aufrufen.

Nachdem das Fenster erstellt ist, können Sie seine Eigenschaften initialisieren und mithilfe der stage-Eigenschaft und Anzeigelistentechniken von Flash Inhalt in das Fenster laden.

In fast allen Fällen sollten Sie die stage-`scaleMode`-Eigenschaft eines neuen nativen Fensters auf `noScale` einstellen (mithilfe der `StageScaleMode.NO_SCALE`-Konstanten). Die Flash-Skalierungsmodi wurden für Situationen konzipiert, in denen dem Anwendungsautor im Voraus das Seitenverhältnis des Anzeigebereichs der Anwendung nicht bekannt ist. Die Skalierungsmodi ermöglichen es dem Autor, den am wenigsten nachteiligen Kompromiss zu wählen: den Inhalt beschneiden, dehnen oder quetschen oder mit leerem Raum auffüllen. Da Sie in AIR den Anzeigebereich (den Fensterrahmen) steuern, können Sie, ohne Kompromisse eingehen zu müssen, die Größe des Fensters an den Inhalt oder den Inhalt an das Fenster anpassen.

Bei Flex- und HTML-Fenstern ist der Skaliermodus automatisch auf `noScale` eingestellt.

Hinweis: *Stellen Sie mithilfe der folgenden statischen NativeWindow-Eigenschaften fest, welche maximalen und minimalen Fenstergrößen unter dem gegenwärtigen Betriebssystem zulässig sind:*

```
var maxOSSize:Point = NativeWindow.systemMaxSize;
var minOSSize:Point = NativeWindow.systemMinSize;
```

Erstellen eines HTML-Fensters

Adobe AIR 1.0 und höher

Sie können ein HTML-Fenster erstellen, indem Sie die `Window.open()`-Methode von JavaScript oder die `createRootWindow()`-Methode der AIR-HTMLLoader-Klasse aufrufen.

Mit HTML-Inhalt in einer Sicherheits-Sandbox können Sie die standardmäßige `Window.open()`-Methode von JavaScript verwenden. Wenn der Inhalt außerhalb von Anwendungs-Sandboxen ausgeführt wird, kann die `open()`-Methode nur auf eine Interaktion des Benutzers, wie etwa einen Mausklick oder einen Tastenanschlag, hin aufgerufen werden. Beim Aufrufen von `open()` wird ein Fenster mit System-Fensterdesign erstellt, indem der Inhalt der angegebenen URL angezeigt wird. Zum Beispiel:

```
newWindow = window.open("xmpl.html", "logWindow", "height=600, width=400, top=10, left=10");
```

Hinweis: *Sie können die HTMLHost-Klasse in ActionScript erweitern, um das mit der `window.open()`-Funktion von JavaScript erstellte Fenster anzupassen. Siehe „Erweitern der HTMLHost-Klasse“ auf Seite 1078.*

Der Inhalt in der Sicherheits-Sandbox der Anwendung kann auf eine leistungsstärkere Methode zur Erstellung von Fenstern, und zwar `HTMLLoader.createRootWindow()`, zugreifen. Mithilfe dieser Methode können Sie alle Erstellungsoptionen für ein neues Fenster festlegen. Im folgenden Beispiel wird mit dem JavaScript-Code ein lightweight-Fenster ohne System-Fensterdesign mit einer Größe von 300 x 400 Pixel erstellt:

```
 var options = new air.NativeWindowInitOptions();
options.systemChrome = "none";
options.type = "lightweight";

var windowBounds = new air.Rectangle(200,250,300,400);
newHTMLLoader = air.HTMLLoader.createRootWindow(true, options, true, windowBounds);
newHTMLLoader.load(new air.URLRequest("xmpl.html"));
```

Hinweis: *Befindet sich der von einem neuen Fenster geladene Inhalt außerhalb der Sicherheits-Sandbox der Anwendung, verfügt das Window-Objekt nicht über die AIR-Eigenschaften `runtime`, `nativeWindow` und `htmlLoader`.*

Wenn Sie ein transparentes Fenster erstellen, wird SWF-Inhalt, der in dem in diesem Fenster geladenen HTML-Code eingebettet ist, nicht immer angezeigt. Sie müssen den `wmode`-Parameter des object- oder embed-Tags, das zum Verweisen auf die SWF-Datei verwendet wird, auf `opaque` oder `transparent` einstellen. Da `wmode` standardmäßig auf `window` eingestellt ist, wird SWF-Inhalt standardmäßig nicht in transparenten Fenstern angezeigt. PDF-Inhalt kann unabhängig vom Wert für `wmode` nicht in transparenten Fenstern angezeigt werden. (Vor AIR 1.5.2 konnte auch SWF-Inhalt nicht in transparenten Fenstern angezeigt werden.)

Fenster, die mithilfe der `createRootWindow()`-Methode erstellt wurden, bleiben unabhängig vom öffnenden Fenster. Die `parent`- und `opener`-Eigenschaften des Window-Objekts von JavaScript lauten `null`. Das öffnende Fenster kann auf das Window-Objekt des neuen Fensters mithilfe der HTMLLoader-Referenz zugreifen, die von der `createRootWindow()`-Funktion zurückgegeben wurde. Im Kontext des vorausgegangenen Beispiels würde die `newHTMLLoader.window`-Anweisung auf das JavaScript-Window-Objekt des erstellten Fensters verweisen.

Hinweis: *Die `createRootWindow()`-Funktion kann sowohl von JavaScript als auch von ActionScript aufgerufen werden.*

Erstellen eines „mx:Window“

Adobe AIR 1.0 und höher

Um ein „mx:Window“ zu erstellen, erstellen Sie eine MXML-Datei, wobei „mx:Window“ als Haupt-Tag dient, oder Sie rufen den Window-Klassenkonstruktor direkt auf.

Im folgenden Beispiel wird ein „mx:Window“ durch Aufrufen des Window-Konstruktors erstellt und angezeigt:

```
var newWindow:Window = new Window();
newWindow.systemChrome = NativeWindowSystemChrome.NONE;
newWindow.transparent = true;
newWindow.title = "New Window";
newWindow.width = 200;
newWindow.height = 200;
newWindow.open(true);
```

Hinzufügen von Inhalt zu einem Fenster

Adobe AIR 1.0 und höher

Wie Sie einem AIR-Fenster Inhalt hinzufügen, hängt vom Fenstertyp ab. Mit MXML und HTML können Sie zum Beispiel den grundlegenden Inhalt des Fensters mit Aussagen definieren. Sie können Ressourcen in den SWF-Dateien der Anwendung einbetten oder Sie können sie aus separaten Anwendungsdateien laden. Flex-, Flash- und HTML-Inhalte können spontan erstellt und einem Fenster dynamisch hinzugefügt werden.

Wenn Sie SWF-Inhalt oder HTML-Inhalt, der JavaScript enthält, laden, müssen Sie das AIR-Sicherheitsmodell berücksichtigen. Sämtlicher Inhalt in der Sicherheits-Sandbox der Anwendung, d. h. Inhalt, der zusammen mir Ihrer Anwendung installiert wird und mit dem app:-URL-Schema geladen werden kann, verfügt über vollständige Zugriffsberechtigung für alle AIR-APIs. Inhalt, der von außerhalb dieser Sandbox geladen wird, kann nicht auf AIR-APIs zugreifen. JavaScript-Inhalt, der sich außerhalb der Anwendungs-Sandbox befindet, kann nicht auf die `runtime`-, `nativeWindow`- oder `htmlLoader`-Eigenschaften des Window-Objekts von JavaScript zugreifen.

Um sicheres Cross-Scripting zu ermöglichen, können Sie mithilfe einer Sandbox Bridge eine begrenzte Schnittstelle zwischen Anwendungsinhalt und anwendungsfremdem Inhalt bereitstellen. Bei HTML-Inhalt können Sie Seiten Ihrer Anwendung einer anwendungsfremden Sandbox zuordnen, um dem Code dieser Seite das Cross-Scripting mit externem Inhalt zu ermöglichen. Siehe „AIR-Sicherheit“ auf Seite 1153.

Laden von SWF-Dateien oder Bildern

Mithilfe der `flash.display.Loader`-Klasse können Sie Flash-SWF-Dateien oder Bilder in die Anzeigeliste eines nativen Fensters laden:

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.display.Loader;

    public class LoadedSWF extends Sprite
    {
        public function LoadedSWF(){
            var loader:Loader = new Loader();
            loader.load(new URLRequest("visual.swf"));
            loader.contentLoaderInfo.addEventListener(Event.COMPLETE,loadFlash);
        }

        private function loadFlash(event:Event):void{
            addChild(event.target.loader);
        }
    }
}
```

Hinweis: *Ältere SWF-Dateien, die mit ActionScript 1 oder 2 erstellt wurden, verwenden gemeinsame Zustände wie Klassendefinitionen, Singletons und globale Variablen, wenn Sie in dasselbe Fenster geladen werden. Wenn eine solche SWF-Datei nur bei unveränderten globalen Zuständen richtig funktioniert, kann sie immer nur einmal in dasselbe Fenster geladen werden. Andernfalls wird sie in dasselbe Fenster als andere SWF-Datei mit überlappenden Klassendefinitionen und Variablen geladen. Dieser Inhalt kann in separate Fenster geladen werden.*

Laden von HTML-Inhalt in ein „NativeWindow“

Es gibt zwei Möglichkeiten, um HTML-Inhalt in ein „NativeWindow“ zu laden. Sie können ein HTMLLoader-Objekt zur Fensterbühne hinzufügen und den HTML-Inhalt in das HTMLLoader-Objekt laden. Mithilfe der `HTMLLoader.createRootWindow()`-Methode können Sie jedoch auch ein Fenster erstellen, das bereits ein HTMLLoader-Objekt enthält. Im folgenden Beispiel wird HTML-Inhalt auf der Bühne eines nativen Fensters in einem Anzeigebereich von 300 x 500 Pixel angezeigt:

```
 //newWindow is a NativeWindow instance
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.width = 300;
htmlView.height = 500;

//set the stage so display objects are added to the top-left and not scaled
newWindow.stage.align = "TL";
newWindow.stage.scaleMode = "noScale";
newWindow.stage.addChild( htmlView );

//urlString is the URL of the HTML page to load
htmlView.load( new URLRequest(urlString) );
```

Mithilfe der Flex-HTML-Komponente können Sie eine HTML-Seite in eine Flex-Anwendung laden.

SWF-Inhalt in HTML-Dateien wird nicht in transparenten Fenstern angezeigt (das heißt, die `transparent`-Eigenschaft des Fensters ist `true`), es sei denn, der `wmode`-Parameter des object- oder embed-Tags, mit dem auf die SWF-Datei verwiesen wird, ist auf `opaque` oder `transparent` eingestellt. Da `wmode` standardmäßig auf `window` eingestellt ist, wird SWF-Inhalt standardmäßig nicht in transparenten Fenstern angezeigt. PDF-Inhalt wird unabhängig vom Wert für `wmode` nicht in einem transparenten Fenster angezeigt.

Außerdem wird weder SWF- noch PDF-Inhalt angezeigt, wenn die HTMLLoader-Steuerung skaliert oder gedreht ist oder wenn die `alpha`-Eigenschaft von HTMLLoader auf einen anderen Wert als 1.0 eingestellt ist.

Hinzufügen von SWF-Inhalt als Überlagerung zu einem HTML-Fenster

Da HTML-Fenster in einer NativeWindow-Instanz enthalten sind, können Sie in der Anzeigeliste Flash-Anzeigobjekte sowohl über als auch unter der HTML-Ebene hinzufügen.

Um ein Anzeigeobjekt über der HTML-Ebene hinzuzufügen, verwenden Sie die `addChild()`-Methode der `window.nativeWindow.stage`-Eigenschaft. Mithilfe der `addChild()`-Methode fügen Sie ebenenbasierten Inhalt über dem im Fenster enthaltenen Inhalt hinzu.

Um ein Anzeigeobjekt unter der HTML-Ebene hinzuzufügen, verwenden Sie die `addChildAt()`-Methode der `window.nativeWindow.stage`-Eigenschaft und übergeben den Wert „0“ für den `index`-Parameter. Wenn Sie ein Objekt an der Indexposition „0“ platzieren, wird der vorhandene Inhalt, einschließlich der HTML-Anzeige, um eine Ebene nach oben verschoben und der neue Inhalt unten eingefügt. Inhalt, der sich in Ebenen unterhalb der HTML-Seite befindet, ist nur sichtbar, wenn Sie die `paintsDefaultBackground`-Eigenschaft des `HTMLlLoader`-Objekts auf `false` einstellen. Darüber hinaus sind alle Elemente der Seite, für die eine Hintergrundfarbe eingestellt ist, nicht transparent. Wenn Sie beispielsweise für das body-Element der Seite eine Hintergrundfarbe einstellen, ist kein Teil der Seite transparent.

Das folgende Beispiel verdeutlicht, wie Sie Flash-Anzeigeobjekte über und unter einer HTML-Seite hinzufügen. In diesem Beispiel werden zwei einfache shape-Objekte erstellt, wobei eines unter und das andere über dem HTML-Inhalt hinzugefügt wird. Außerdem wird anhand des `enterFrame`-Ereignisses die shape-Position aktualisiert.

```
<html>
<head>
<title>Bouncers</title>
<script src="AIRAliases.js" type="text/javascript"></script>
<script language="JavaScript" type="text/javascript">
air.Shape = window.runtime.flash.display.Shape;

function Bouncer(radius, color){
    this.radius = radius;
    this.color = color;

    //velocity
    this.vX = -1.3;
    this.vY = -1;

    //Create a Shape object and draw a circle with its graphics property
    this.shape = new air.Shape();
    this.shape.graphics.lineStyle(1,0);
    this.shape.graphics.beginFill(this.color,.9);
    this.shape.graphics.drawCircle(0,0,this.radius);
    this.shape.graphics.endFill();

    //Set the starting position
    this.shape.x = 100;
    this.shape.y = 100;


    //Moves the sprite by adding (vX,vY) to the current position
    this.update = function(){
        this.shape.x += this.vX;
        this.shape.y += this.vY;

        //Keep the sprite within the window
        if( this.shape.x - this.radius < 0){
            this.vX = -this.vX;
        }
        if( this.shape.y - this.radius < 0){
            this.vY = -this.vY;
        }
        if( this.shape.x  + this.radius > window.nativeWindow.stage.stageWidth){
            this.vX = -this.vX;
        }
        if( this.shape.y  + this.radius > window.nativeWindow.stage.stageHeight){
            this.vY = -this.vY;
        }

    };
}

function init(){
    //turn off the default HTML background
    window.htmlLoader.paintsDefaultBackground = false;
    var bottom = new Bouncer(60,0xff2233);
```

```
    var top = new Bouncer(30,0x2441ff);

    //listen for the enterFrame event
    window.htmlLoader.addEventListener("enterFrame",function(evt){
        bottom.update();
        top.update();
    });

    //add the bouncing shapes to the window stage
    window.nativeWindow.stage.addChildAt(bottom.shape,0);
    window.nativeWindow.stage.addChild(top.shape);
}
</script>
<body onload="init();">
<h1>de Finibus Bonorum et Malorum</h1>
<p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis
et quasi architecto beatae vitae dicta sunt explicabo.</p>
<p style="background-color:#FFFF00; color:#660000;">This paragraph has a background color.</p>
<p>At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis
praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias
excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui
officia deserunt mollitia animi, id est laborum et dolorum fuga.</p>
</body>
</html>
```

Dieses Beispiel dient als rudimentäre Einführung in einige erweiterte Techniken, die die Grenzen zwischen JavaScript und ActionScript in AIR überschreiten. Wenn Sie mit der Verwendung von ActionScript-Anzeigeobjekten nicht vertraut sind, lesen Sie den Abschnitt „Programmieren von Anzeigeobjekten“ auf Seite 161 im *ActionScript 3.0 Entwicklerhandbuch*.

Beispiel: Erstellen eines nativen Fensters

Adobe AIR 1.0 und höher

Im folgenden Beispiel wird verdeutlicht, wie Sie ein natives Fenster erstellen:

```
public function createNativeWindow():void {
    //create the init options
    var options:NativeWindowInitOptions = new NativeWindowInitOptions();
    options.transparent = false;
    options.systemChrome = NativeWindowSystemChrome.STANDARD;
    options.type = NativeWindowType.NORMAL;

    //create the window
    var newWindow:NativeWindow = new NativeWindow(options);
    newWindow.title = "A title";
    newWindow.width = 600;
    newWindow.height = 400;

    newWindow.stage.align = StageAlign.TOP_LEFT;
    newWindow.stage.scaleMode = StageScaleMode.NO_SCALE;

    //activate and show the new window
    newWindow.activate();
}
```

Verwalten von Fenstern

Adobe AIR 1.0 und höher

Mithilfe der Eigenschaften und Methoden der NativeWindow-Klasse verwalten Sie das Erscheinungsbild, das Verhalten und den Lebenszyklus von Desktopfenstern.

Hinweis: *Wenn Sie mit dem Flex-Framework arbeiten, ist es im Allgemeinen besser, das Fensterverhalten mithilfe der Framework-Klassen zu verwalten. Auf die meisten NativeWindow-Eigenschaften und -Methoden können Sie über die mx:WindowedApplication- und mx:Window-Klasse zugreifen.*

Abrufen einer NativeWindow-Instanz

Adobe AIR 1.0 und höher

Um ein Fenster bearbeiten zu können, müssen Sie zuerst die Window-Instanz abrufen. Eine Window-Instanz können Sie von einem der folgenden Orte abrufen:

- Der native Fensterkonstruktor, der zum Erstellen des Fensters verwendet wird:

  ```
  var win:NativeWindow = new NativeWindow(initOptions);
  ```

- Die `nativeWindow`-Eigenschaft der Fensterbühne:

  ```
  var win:NativeWindow = stage.nativeWindow;
  ```

- Die `stage`-Eigenschaft eines Anzeigeobjekts im Fenster:

  ```
  var win:NativeWindow = displayObject.stage.nativeWindow;
  ```

- Die `target`-Eigenschaft eines nativen Fensterereignisses, das vom Fenster ausgelöst wird:

  ```
  private function onNativeWindowEvent(event:NativeWindowBoundsEvent):void
  {
      var win:NativeWindow = event.target as NativeWindow;
  }
  ```

- Die `nativeWindow`-Eigenschaft einer HTML-Seite, die im Fenster angezeigt wird:

  ```
  var win:NativeWindow = htmlLoader.window.nativeWindow;
  ```

- Die `activeWindow`- und `openedWindows`-Eigenschaften des NativeApplication-Objekt:

  ```
  var nativeWin:NativeWindow = NativeApplication.nativeApplication.activeWindow;
  var firstWindow:NativeWindow = NativeApplication.nativeApplication.openedWindows[0];
  ```

  `NativeApplication.nativeApplication.activeWindow` verweist auf das aktive Fenster einer Anwendung (gibt jedoch `null` zurück, wenn es sich bei dem aktiven Fenster nicht um ein Fenster dieser AIR-Anwendung handelt). Das `NativeApplication.nativeApplication.openedWindows`-Array enthält alle Fenster in einer AIR-Anwendung, die nicht geschlossen wurden.

Da es sich bei den Flex-Objekten mx:WindowedApplication und mx:Window um Anzeigeobjekte handelt, können Sie in einer MXML-Datei mithilfe der `stage`-Eigenschaft wie folgt leicht auf das Anwendungsfenster verweisen:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
applicationComplete="init();">
    <mx:Script>
        <![CDATA[
            import flash.display.NativeWindow;

            public function init():void{
                var appWindow:NativeWindow = this.stage.nativeWindow;
                //set window properties
                appWindow.visible = true;
            }
        ]]>
    </mx:Script>
</WindowedApplication
```

Hinweis: *Solange die WindowedApplication- oder Window-Komponente noch nicht vom Flex-Framework zur Fensterbühne hinzugefügt wurde, lautet die `stage`-Eigenschaft der Komponente `null`. Dieses Verhalten entspricht dem der Flex Application-Komponente. Es bedeutet jedoch, dass Sie nicht auf die Bühne oder die NativeWindow-Instanz in Ereignis-Listenern zugreifen können, die zu einem früheren Zeitpunkt im Initialisierungszyklus der WindowedApplication- und Window-Komponenten auftreten, wie etwa `creationComplete`. Nachdem das `applicationComplete`-Ereignis ausgelöst wurde, können Sie problemlos auf die Bühne und die NativeWindow-Instanz zugreifen.*

Aktivieren, Anzeigen und Ausblenden von Fenstern

Adobe AIR 1.0 und höher

Sie aktivieren ein Fenster, indem Sie die `activate()`-Methode von NativeWindow aufrufen. Durch das Aktivieren wird das Fenster in den Vordergrund gestellt, es erhält den Tastatur- und Mausfokus und wird, bei Bedarf, sichtbar gemacht, indem das Fenster wiederhergestellt oder die `visible`-Eigenschaft auf `true` eingestellt wird. Durch das Aktivieren eines Fensters wird die Reihenfolge der anderen Fenster in der Anwendung nicht verändert. Durch das Aufrufen der `activate()`-Methode wird das Fenster veranlasst, ein `activate`-Ereignis auszulösen.

Um ein Fenster einzublenden. ohne es zu aktivieren, stellen Sie seine `visible`-Eigenschaft auf `true` ein. Dadurch wird das Fenster in den Vordergrund gestellt, erhält aber nicht den Fokus.

Um ein Fenster auszublenden, stellen Sie seine `visible`-Eigenschaft auf `false` ein. Durch das Ausblenden des Fensters wird die Anzeige des Fensters und dazugehöriger Taskleisten-Symbole, sowie, unter Mac OS X, der Eintrag im Windows-Menü unterdrückt.

Wenn Sie die Sichtbarkeit eines Fensters ändern, ändert sich auch die Sichtbarkeit der Fenster, die sich im Besitz des jeweiligen Fensters befinden. Wenn Sie beispielsweise ein Fenster ausblenden, werden auch alle Fenster, die dieses Fenster besitzt, ausgeblendet.

Hinweis: *Unter Mac OS X ist es nicht möglich, ein minimiertes Fenster vollständig zu verbergen, das über ein Symbol im Fensterbereich des Docks verfügt. Ist bei einem minimierten Fenster die `visible`-Eigenschaft auf `false` eingestellt, wird das Dock-Symbol des Fensters weiterhin angezeigt. Durch Klicken auf das Symbol wird das Fenster wieder angezeigt.*

Ändern der Anzeigereihenfolge von Fenstern

Adobe AIR 1.0 und höher

AIR bietet mehrere Methoden, mit denen die Anzeigereihenfolge der Fernster direkt geändert werden kann. Sie können ein Fenster an den Anfang oder das Ende der Anzeigereihenfolge verschieben, und Sie können ein Fenster vor oder hinter ein anderes Fenster verschieben. Außerdem können Sie die Anzeigereihenfolge von Fenstern ändern, indem Sie sie aktivieren.

Sie können dafür sorgen, dass ein Fenster immer vor anderen Fenstern angezeigt wird, indem Sie seine `alwaysInFront`-Eigenschaft auf `true` einstellen. Wenn mehrere Fenster diese Einstellung aufweisen, wird eine Anzeigereihenfolge unter diesen Fenstern erstellt, sie werden aber immer vor Fenstern angezeigt, deren `alwaysInFront`-Eigenschaft auf „false“ eingestellt ist.

Fenster in der obersten Gruppe werden außerdem vor Fenstern in anderen Anwendungen angezeigt, selbst wenn die AIR-Anwendung nicht aktiv ist. Da sich dieses Verhalten störend auf den Benutzer auswirken kann, sollte `alwaysInFront` nur auf `true` eingestellt werden, wenn es erforderlich und angemessen ist. Beispiele:

- Temporäre Popupfenster für Steuereinheiten, wie etwa QuickInfos, Popuplisten, benutzerdefinierte Menüs oder Kombinationsfelder. Da diese Fenster geschlossen werden, wenn sie den Fokus verlieren, wird Benutzern nicht die Sicht auf andere Fenster genommen und sie fühlen sich nicht gestört.
- Sehr dringende Fehlermeldungen und Warnungen. Wenn eine unwiderrufliche Änderung auftreten kann, falls der Benutzer nicht rechtzeitig reagiert, kann es gerechtfertigt sein, eine Warnmeldung im Vordergrund einzublenden. Die meisten Fehlermeldungen und Warnungen können jedoch in der normalen Anzeigereihenfolge der Fenster verarbeitet werden.
- Kurzfristige überlappende Fenster

Hinweis: *AIR erzwingt nicht die korrekte Verwendung der `alwaysInFront`-Eigenschaft. Wenn der Workflow eines Benutzers jedoch von Ihrer Anwendung unterbrochen wird, wird sie wahrscheinlich dem Papierkorb dieses Benutzers übergeben.*

Wenn ein Fenster andere Fenster besitzt, werden diese Fenster immer vor dem übergeordneten Fenster angeordnet. Wenn Sie für ein Fenster, das andere Fenster besitzt, `orderToFront()` aufrufen oder `alwaysInFront` auf `true` einstellen, werden die untergeordneten Fenster zusammen mit dem Besitzerfenster vor anderen Fenstern angeordnet, aber die untergeordneten Fenster werden immer vor dem Besitzerfenster angezeigt.

Das Aufrufen von Methoden zur Fensteranordnung für Fenster, die sich im Besitz eines anderen Fensters befinden, funktioniert normal, vorausgesetzt, alle Fenster haben denselben Besitzer; es ist jedoch möglich, dass sich die Anordnung aller untergeordneten Fenster relativ zu den Fenstern ändert, die nicht zur Gruppe gehören. Wenn Sie beispielsweise für ein Fenster, das sich im Besitz eines anderen Fensters befindet, `orderToFront()` aufrufen, wird dieses Fenster zusammen mit seinem Besitzerfenster und allen anderen Fenstern, die denselben Besitzer haben, in der Reihenfolge der Fensteranzeige ganz nach vorne verschoben.

Die NativeWindow-Klasse bietet die folgenden Eigenschaften und Methoden, mit denen die Anzeigereihenfolge eines Fensters relativ zu anderen Fenstern festgelegt werden kann:

Mitglied	Beschreibung
alwaysInFront-Eigenschaft	Gibt an, ob das Fenster in der obersten Fenstergruppe angezeigt wird. In fast allen Fällen ist `false` die beste Einstellung. Durch eine Änderung des Werts von `false` in `true` wird das Fenster vor allen anderen Fenstern angezeigt (es wird jedoch nicht aktiviert). Beim Ändern des Werts von `true` in `false` wird das Fenster hinter die anderen Fenster in der obersten Gruppe gestellt, es bleibt jedoch vor anderen Fenstern. Wenn die Eigenschaft auf ihren aktuellen Wert eingestellt wird, ändert sich die Anzeigereihenfolge der Fenster nicht. Die `alwaysInFront`-Einstellung hat keine Auswirkung auf Fenster, die sich im Besitz eines anderen Fensters befinden.
orderToFront()	Stellt das Fenster in den Vordergrund.
orderInFrontOf()	Stellt das Fenster direkt vor ein bestimmtes Fenster.
orderToBack()	Stellt das Fenster hinter andere Fenster.
orderBehind()	Stellt das Fenster direkt hinter ein bestimmtes Fenster.
activate()	Stellt das Fenster in den Vordergrund (blendet es ein und weist ihm den Fokus zu).

Hinweis: *Das Aufrufen der Methoden für die Anzeigereihenfolge hat keine Wirkung, wenn ein Fenster ausgeblendet (der Wert für `visible` lautet `false`) oder minimiert ist.*

Unter dem Linux-Betriebssystem erzwingen verschiedene Fenstermanager unterschiedliche Regeln bezüglich der Anzeigereihenfolge der Fenster:

- Bei einigen Fenstermanagern werden Dienstprogrammfenster vor normalen Fenstern angezeigt.
- Bei bestimmten Fenstermanagern wird ein Vollbildfenster, dessen `alwaysInFront`-Eigenschaft auf `true` eingestellt ist, immer vor anderen Fenstern angezeigt, bei denen `alwaysInFront` ebenfalls auf `true` eingestellt ist.

Schließen von Fenstern

Adobe AIR 1.0 und höher

Sie können ein Fenster mithilfe der `NativeWindow.close()`-Methode schließen.

Beim Schließen eines Fensters werden die Inhalte des Fenster entfernt, wenn jedoch andere Objekte auf diesen Inhalt verweisen, werden die Inhaltsobjekte nicht gelöscht. Die `NativeWindow.close()`-Methode wird asynchron ausgeführt, und die im Fenster enthaltene Anwendung wird während des Schließvorgangs ausgeführt. Die close-Methode löst ein close-Ereignis aus, wenn der Schließvorgang abgeschlossen ist. Das NativeWindow-Objekt ist im Prinzip noch gültig, aber der Zugriff auf die meisten Eigenschaften und Methoden eines geschlossenen Fensters erzeugt einen IllegalOperationError. Ein geschlossenes Fenster kann nicht erneut geöffnet werden. Überprüfen Sie die `closed`-Eigenschaft eines Fensters, um zu prüfen, ob das Fenster geschlossen wurde. Um ein Fenster einfach nur auszublenden, stellen Sie seine `NativeWindow.visible`-Eigenschaft auf `false` ein.

Ist die `Nativeapplication.autoExit`-Eigenschaft auf `true` eingestellt (die Standardeinstellung), wird die Anwendung mit dem Schließen ihres letzten Fensters beendet.

Jedes Fenster, das sich im Besitz eines anderen Fensters befindet, wird zusammen mit seinem Besitzerfenster geschlossen. Die Fenster, die sich im Besitz eines anderen Fensters befinden, lösen kein closing-Ereignis aus und können das Schließen daher nicht verhindern. Ein close-Ereignis wird ausgelöst.

Abbrechen von Fenstervorgängen ermöglichen

Adobe AIR 1.0 und höher

Wurde einem Fenster das System-Fensterdesign zugewiesen, können Benutzerinteraktionen mit dem Fenster abgebrochen werden, indem auf Standardverhalten der entsprechenden Ereignisse gewartet wird, um sie anschließend abzubrechen. Wenn beispielsweise ein Benutzer auf die Schaltfläche „Schließen“ des System-Fensterdesigns klickt, wird das `closing`-Ereignis ausgelöst. Wird die `preventDefault()`-Methode des Ereignisses durch einen registrierten Listener aufgerufen, wird das Fenster nicht geschlossen.

Wenn einem Fenster nicht das System-Fensterdesign zugewiesen wurde, werden Benachrichtigungsereignisse für beabsichtigte Änderungen nicht automatisch vor der Änderung ausgelöst. Das bedeutet, wenn die Methoden zum Schließen eines Fensters oder Ändern des Fensterzustands aufgerufen werden, oder eine der bound-Eigenschaften des Fensters einstellt wird, kann die Änderung nicht abgebrochen werden. Die Anwendungslogik kann das relevante Benachrichtigungsereignis mithilfe der `dispatchEvent()`-Methode des Fensters auslösen, um Komponenten in Ihrer Anwendung auf bevorstehende Fensteränderungen hinzuweisen.

Im folgenden Beispiel wird eine abbrechbare Ereignisprozedur für die Schaltfläche „Schließen“ eines Fensters implementiert:

```
public function onCloseCommand(event:MouseEvent):void{
    var closingEvent:Event = new Event(Event.CLOSING,true,true);
    dispatchEvent(closing);
    if(!closingEvent.isDefaultPrevented()){
        win.close();
    }
}
```

Die `dispatchEvent()`-Methode gibt `false` zurück, wenn die `preventDefault()`-Ereignismethode durch einen Listener aufgerufen wird. Da sie jedoch auch aus anderen Gründen `false` zurückgeben kann, empfiehlt es sich, mithilfe der `isDefaultPrevented()`-Methode ausdrücklich zu prüfen, ob die Änderungen abgebrochen werden sollen.

Maximieren, Minimieren und Wiederherstellen von Fenstern

Adobe AIR 1.0 und höher

Sie können ein Fenster mithilfe der `maximize()`-Methode der NativeWindow-Klasse maximieren.

```
myWindow.maximize();
```

Sie können ein Fenster mithilfe der `minimize()`-Methode der NativeWindow-Klasse minimieren.

```
myWindow.minimize();
```

Mithilfe der `restore()`-Methode der NativeWindow-Klasse können Sie ein Fenster wiederherstellen, d. h. die Größe auf den Wert vor dem Minimieren oder Maximieren zurücksetzen.

```
myWindow.restore();
```

Ein Fenster, das sich im Besitz eines anderen Fensters befindet, wird zusammen mit dem Besitzerfenster minimiert und wiederhergestellt. Wenn ein Fenster, das sich im Besitz eines anderen Fensters befindet, zusammen mit dem Besitzerfenster minimiert wird, löst es keine Ereignisse aus.

Hinweis: *Das aus dem Maximieren eines AIR-Fensters resultierende Verhalten unterscheidet sich vom Standardverhalten unter Mac OS X. Statt zwischen einer durch die Anwendung definierten „Standardgröße" und der letzten durch den Benutzer eingestellten Größe umzuschalten, wird bei AIR-Fenstern zwischen der zuletzt durch die Anwendung oder den Benutzer eingestellten Größe und dem gesamten verwendbaren Bildschirmbereich umgeschaltet.*

Unter dem Linux-Betriebssystem erzwingen verschiedene Fenstermanager unterschiedliche Regeln bezüglich der Einstellung des Anzeigestatus der Fenster:

- Unter einigen Fenstermanagern können Dienstprogrammfenster nicht maximiert werden.
- Wenn für ein Fenster eine maximale Größe festgelegt wurde, lassen einige Fenstermanager die Maximierung des Fensters nicht zu. Andere Fenstermanager maximieren den Anzeigestatus, ändern aber nicht die Größe des Fensters. In beiden Fällen wird kein Ereignis für die Anzeigestatusänderung ausgelöst.
- Einige Fenstermanager berücksichtigen die `maximizable`- oder `minimizable`-Einstellungen des Fensters nicht.

Hinweis: *Unter Linux werden Fenstereigenschaften asynchron geändert. Wenn Sie den Anzeigestatus in einer Programmzeile ändern und den Wert in der nächsten Zeile lesen, gibt der gelesene Wert immer noch die alte Einstellung an. Auf allen Plattformen löst das NativeWindow-Objekt das `displayStateChange`-Ereignis aus, wenn sich der Anzeigestatus ändert. Wenn Sie Aktionen ausführen müssen, die auf dem neuen Status des Fensters basieren, verwenden Sie dazu eine `displayStateChange`-Ereignisprozedur. Lesen Sie dazu „Warten auf Window-Ereignisse" auf Seite 976.*

Beispiel: Minimieren, Maximieren, Wiederherstellen und Schließen von Fenstern

Adobe AIR 1.0 und höher

Anhand der folgenden kurzen MXML-Anwendung werden die `maximize()`-, `minimize()`-, `restore()`- und `close()`-Methoden der Window-Klasse verdeutlicht:

```
<?xml version="1.0" encoding="utf-8"?>

<mx:WindowedApplication
    xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical">


    <mx:Script>
    <![CDATA[
    public function minimizeWindow():void
    {
        this.stage.nativeWindow.minimize();
    }

    public function maximizeWindow():void
    {
        this.stage.nativeWindow.maximize();
    }

    public function restoreWindow():void
    {
        this.stage.nativeWindow.restore();
    }

    public function closeWindow():void
    {
        this.stage.nativeWindow.close();
    }
    ]]>
    </mx:Script>

    <mx:VBox>
        <mx:Button label="Minimize" click="minimizeWindow()"/>
        <mx:Button label="Restore" click="restoreWindow()"/>
        <mx:Button label="Maximize" click="maximizeWindow()"/>
        <mx:Button label="Close" click="closeWindow()"/>
    </mx:VBox>

</mx:WindowedApplication>
```

Im folgenden ActionScript-Beispiel für Flash werden vier klickbare Textfelder erstellt, mit denen die `minimize()`-, `maximize()`-, `restore()`- und `close()`-Methoden der NativeWindow-Klasse ausgelöst werden:

```
package
{
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.text.TextField;

    public class MinimizeExample extends Sprite
    {
        public function MinimizeExample():void
        {
            var minTextBtn:TextField = new TextField();
            minTextBtn.x = 10;
            minTextBtn.y = 10;
            minTextBtn.text = "Minimize";
            minTextBtn.background = true;
            minTextBtn.border = true;
            minTextBtn.selectable = false;
            addChild(minTextBtn);
            minTextBtn.addEventListener(MouseEvent.CLICK, onMinimize);

            var maxTextBtn:TextField = new TextField();
            maxTextBtn.x = 120;
            maxTextBtn.y = 10;
            maxTextBtn.text = "Maximize";
            maxTextBtn.background = true;
            maxTextBtn.border = true;
            maxTextBtn.selectable = false;
            addChild(maxTextBtn);
            maxTextBtn.addEventListener(MouseEvent.CLICK, onMaximize);

            var restoreTextBtn:TextField = new TextField();
            restoreTextBtn.x = 230;
            restoreTextBtn.y = 10;
            restoreTextBtn.text = "Restore";
            restoreTextBtn.background = true;
            restoreTextBtn.border = true;
            restoreTextBtn.selectable = false;
            addChild(restoreTextBtn);
            restoreTextBtn.addEventListener(MouseEvent.CLICK, onRestore);

            var closeTextBtn:TextField = new TextField();
            closeTextBtn.x = 340;
            closeTextBtn.y = 10;
            closeTextBtn.text = "Close Window";
            closeTextBtn.background = true;
            closeTextBtn.border = true;
            closeTextBtn.selectable = false;
            addChild(closeTextBtn);
```

```
            closeTextBtn.addEventListener(MouseEvent.CLICK, onCloseWindow);
        }
        function onMinimize(event:MouseEvent):void
        {
            this.stage.nativeWindow.minimize();
        }
        function onMaximize(event:MouseEvent):void
        {
            this.stage.nativeWindow.maximize();
        }
        function onRestore(event:MouseEvent):void
        {
            this.stage.nativeWindow.restore();
        }
        function onCloseWindow(event:MouseEvent):void
        {
            this.stage.nativeWindow.close();
        }
    }
}
```

Skalieren und Verschieben von Fenstern

Adobe AIR 1.0 und höher

Wurde einem Fenster das System-Fensterdesign zugewiesen, stehen dadurch Haltegriffe zur Verfügung, um durch Ziehen die Größe des Fensters zu skalieren und um es auf dem Desktop zu verschieben. Wenn einem Fenster nicht das System-Fensterdesign zugewiesen wurde, müssen Sie selbst dem Fenster Steuerelemente hinzufügen, um dem Benutzer das Skalieren und Verschieben des Fensters zu ermöglichen.

Hinweis: *Um ein Fenster zu skalieren oder zu verschieben, müssen Sie zuerst einen Verweis auf die NativeWindow-Instanz abrufen. Weitere Informationen zum Abrufen einer Fensterreferenz finden Sie unter „Abrufen einer NativeWindow-Instanz“ auf Seite 965.*

Skalieren von Fenstern

Damit ein Benutzer die Größe eines Fensters interaktiv ändern kann, verwenden Sie die NativeWindow-Methode `startResize()`. Wenn diese Methode durch ein `mouseDown`-Ereignis aufgerufen wird, wird der Skalierungsvorgang von der Maus bestimmt. Er wird beendet, wenn das Betriebssystem ein `mouseUp`-Ereignis empfängt. Beim Aufrufen von `startResize()` wird ein Argument übergeben, das die Kante oder die Ecke angibt, ab der das Fenster skaliert wird.

Um die Fenstergröße programmgesteuert festzulegen, stellen Sie die Eigenschaften `width`, `height` oder `bounds` des Fensters auf die gewünschten Größen. Wenn Sie die Ränder festlegen, können die Fenstergröße und -position gleichzeitig geändert werden. Die Reihenfolge, in der die Änderungen wirksam werden, ist jedoch nicht gewährleistet. Einige Linux-Fenstermanager lassen nicht zu, dass sich Fenster über die Grenzen des Desktopbildschirms hinaus erstrecken. In diesen Fällen kann die endgültige Fenstergröße aufgrund der Reihenfolge, in der die Eigenschaften festgelegt werden, beschränkt sein, selbst wenn der tatsächliche Effekt der Änderungen andernfalls zu einem zulässigen Fenster geführt hätte. Wenn Sie zum Beispiel Höhe und Position eines Fensters im unteren Bereich des Bildschirms ändern, wird die vollständige Höhenänderung ggf. nicht umgesetzt, wenn die Höhenänderung vor der Änderung der y-Position vorgenommen wird.

Hinweis: *Unter Linux werden Fenstereigenschaften asynchron geändert. Wenn Sie die Größe eines Fensters in einer Zeile Ihres Programms ändern und dann in der nächsten Zeile die Abmessungen lesen, stellen sie immer noch die alten Einstellungen dar. Auf allen Plattformen löst das NativeWindow-Objekt das* `resize`*-Ereignis aus, wenn die Größe des Fensters geändert wird. Wenn Sie Aktionen ausführen müssen, die auf der neuen Größe oder dem neuen Status des Fensters basieren, zum Beispiel Anordnen der Steuerungen im Fenster, verwenden Sie dazu immer eine* `resize`*-Ereignisprozedur. Lesen Sie dazu „Warten auf Window-Ereignisse“ auf Seite 976.*

Der Skaliermodus der Bühne bestimmt, wie sich die Fensterbühne und ihr Inhalt bei einer Skalierung des Fensters verhält. Beachten Sie, dass die Skaliermodi der Bühne für Situationen konzipiert sind, wie etwa einen Webbrowser, in der die Anwendung nicht die Größe oder das Seitenverhältnis des Anzeigebereichs steuert. Im Allgemeinen erhalten Sie die besten Ergebisse, wenn Sie die `scaleMode`-Eigenschaft der Bühne auf `StageScaleMode.NO_SCALE` einstellen. Wenn der Inhalt des Fensters skaliert werden soll, können Sie die `scaleX`- und `scaleY`-Parameter des Inhalts als Reaktion auf die Änderungen der Fenstergrenzen einstellen.

Verschieben von Fenstern

Mithilfe der NativeWindow-Methode `startMove()` verschieben Sie ein Fenster, ohne es zu skalieren. Wie bei der `startResize()`-Methode wird beim Aufrufen der `startMove()`-Methode durch ein `mouseDown`-Ereignis der Verschiebungsvorgang von der Maus bestimmt. Er wird beendet, wenn das Betriebssystem ein `mouseUp`-Ereignis empfängt.

Weitere Informationen über die `startResize()`- und `startMove()`-Methoden finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Um ein Fenster programmgesteuert zu verschieben, stellen Sie die `x`-, `y`- oder `bounds`-Eigenschaften des Fensters auf die gewünschten Werte ein. Wenn Sie die Ränder (bounds) festlegen, können die Fenstergröße und -position gleichzeitig geändert werden.

Hinweis: *Unter Linux werden Fenstereigenschaften asynchron geändert. Wenn Sie ein Fenster in einer Programmzeile verschieben und die Position in der nächsten Zeile lesen, gibt der gelesene Wert immer noch die alte Einstellung an. Auf allen Plattformen löst das NativeWindow-Objekt das* `move`*-Ereignis aus, wenn sich die Position ändert. Wenn Sie Aktionen ausführen müssen, die auf der neuen Position des Fensters basieren, verwenden Sie dazu eine* `move`*-Ereignisprozedur. Lesen Sie dazu „Warten auf Window-Ereignisse“ auf Seite 976.*

Beispiel: Skalieren und Verschieben von Fenstern

Adobe AIR 1.0 und höher

Im folgenden Beispiel wird gezeigt, wie Skalierungs- und Verschiebungsvorgänge an einem Fenster eingeleitet werden:

```
package
{
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.display.NativeWindowResize;

    public class NativeWindowResizeExample extends Sprite
    {
        public function NativeWindowResizeExample():void
        {
            // Fills a background area.
            this.graphics.beginFill(0xFFFFFF);
            this.graphics.drawRect(0, 0, 400, 300);
            this.graphics.endFill();

            // Creates a square area where a mouse down will start the resize.
            var resizeHandle:Sprite =
                createSprite(0xCCCCCC, 20, this.width - 20, this.height - 20);
            resizeHandle.addEventListener(MouseEvent.MOUSE_DOWN, onStartResize);

            // Creates a square area where a mouse down will start the move.
            var moveHandle:Sprite = createSprite(0xCCCCCC, 20, this.width - 20, 0);
            moveHandle.addEventListener(MouseEvent.MOUSE_DOWN, onStartMove);
        }

        public function createSprite(color:int, size:int, x:int, y:int):Sprite
        {
            var s:Sprite = new Sprite();
            s.graphics.beginFill(color);
            s.graphics.drawRect(0, 0, size, size);
            s.graphics.endFill();
            s.x = x;
            s.y = y;
            this.addChild(s);
            return s;
        }

        public function onStartResize(event:MouseEvent):void
        {
            this.stage.nativeWindow.startResize(NativeWindowResize.BOTTOM_RIGHT);
        }

        public function onStartMove(event:MouseEvent):void
        {
            this.stage.nativeWindow.startMove();
        }
    }
}
```

Warten auf Window-Ereignisse

Adobe AIR 1.0 und höher

Registrieren Sie einen Listener mit der Window-Instanz, um auf Ereignisse zu warten, die von einem Fenster ausgelöst werden. Um beispielsweise auf das closing-Ereignis zu warten, registrieren Sie einen Listener wie folgt mit dem Fenster:

```
myWindow.addEventListener(Event.CLOSING, onClosingEvent);
```

Wenn ein Ereignis ausgelöst wird, verweist die `target`-Eigenschaft auf das Fenster, das das Ereignis ausgelöst hat.

Die meisten Window-Ereignisse weisen zwei miteinander verbundene Meldungen auf. Die erste Meldung weist darauf hin, dass eine Fensteränderung bevorsteht (und abgebrochen werden kann), und die zweite Meldung gibt an, dass die Änderung erfolgt ist. Wenn beispielsweise ein Benutzer auf die Schaltfläche „Schließen“ eines Fensters klickt, wird die closing-Ereignismeldung ausgelöst. Wird das Ereignis nicht durch einen Listener abgebrochen, wird das Fenster geschlossen und das close-Ereignis wird ausgelöst.

In der Regel werden warnende Ereignisse, wie etwa `closing`, nur ausgelöst, wenn ein Ereignis mithilfe des System-Fensterdesigns ausgelöst wurde. Beispielsweise wird beim Aufrufen der `close()`-Methode eines Fensters nicht automatisch das `closing`-Ereignis ausgelöst - nur das `close`-Ereignis wird ausgelöst. Sie können jedoch ein closing-Ereignisobjekt erstellen und es mithilfe der `dispatchEvent()`-Methode des Fensters auslösen.

Die window-Ereignisse, die ein Event-Objekt auslösen, sind:

Ereignis	Beschreibung
activate	Wird ausgelöst, wenn das Fenster den Fokus erhält.
deactivate	Wird ausgelöst, wenn das Fenster den Fokus verliert.
closing	Wird ausgelöst, kurz bevor das Fenster geschlossen wird. Erfolgt nur automatisch, wenn die Schaltfläche „Schließen“ des System-Fensterdesigns angeklickt oder, unter Mac OS X, der Befehl „Beenden“ aufgerufen wird.
close	Wird ausgelöst, wenn das Fenster geschlossen wurde.

Die window-Ereignisse, die ein NativeBoundsEvent-Objekt auslösen, sind:

Ereignis	Beschreibung
moving	Wird ausgelöst, kurz bevor die Position der linken oberen Ecke des Fensters geändert wird, entweder da das Fenster verschoben oder skaliert oder der Anzeigezustand des Fensters geändert wird.
move	Wird ausgelöst, nachdem die Position der linken oberen Ecke des Fensters geändert wurde.
resizing	Wird ausgelöst, kurz bevor die Breite oder Höhe des Fensters geändert wird, entweder da das Fenster skaliert oder der Anzeigezustand des Fensters geändert wird.
resize	Wird ausgelöst, nachdem die Größe des Fensters geändert wurde.

Bei NativeWindowBoundsEvent-Ereignissen können Sie mithilfe der `beforeBounds`- und `afterBounds`-Ereignisse die Grenzen des Fensters vor und nach der bevorstehenden oder abgeschlossenen Änderung bestimmen.

Die window-Ereignisse, die ein NativeWindowDisplayStateEvent-Objekt auslösen, sind:

Ereignis	Beschreibung
displayStateChanging	Wird ausgelöst, kurz bevor der Anzeigezustand des Fensters geändert wird.
displayStateChange	Wird ausgelöst, nachdem der Anzeigezustand des Fensters geändert wurde.

Bei NativeWindowDisplayStateEvent-Ereignissen können Sie mithilfe der `beforeDisplayState-` und `afterDisplayState`-Ereignisse den Anzeigezustand des Fensters vor und nach der bevorstehenden oder abgeschlossenen Änderung bestimmen.

Unter einigen Linux-Fenstermanagern wird kein Anzeigestatusänderungs-Ereignis ausgelöst, wenn ein Fenster mit einer Einstellung für die maximale Größe maximiert wird. (Für das Fenster wird der maximierte Anzeigestatus eingestellt, seine Größe wird jedoch nicht geändert.)

Anzeigen von Fenstern im Vollbildmodus

Adobe AIR 1.0 und höher

Durch das Einstellen der `displayState`-Eigenschaft der Bühne auf `StageDisplayState.FULL_SCREEN_INTERACTIVE` wird das Fenster im Vollbildmodus angezeigt; in diesem Modus sind Eingaben über die Tastatur *zulässig*. (Bei SWF-Inhalt, der in einem Browser ausgeführt wird, ist die Eingabe über die Tastatur nicht zulässig). Durch Drücken der Esc-Taste wird der Vollbildmodus beendet.

Hinweis: *Unter einigen Linux-Fenstermanagern werden die Fensterabmessungen nicht geändert, um den Bildschirm zu füllen, wenn für das Fenster eine maximale Größe festgelegt wurde (das System-Fensterdesign wird jedoch entfernt).*

Im folgenden Beispiel definiert der Flex-Code eine einfache AIR-Anwendung, mit der ein einfaches Terminal im Vollbildmodus eingerichtet wird:

```
 <?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    applicationComplete="init()" backgroundColor="0x003030" focusRect="false">
    <mx:Script>
        <![CDATA[
            private function init():void
            {
                stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
                focusManager.setFocus(terminal);
                terminal.text = "Welcome to the dumb terminal app. Press the ESC key to
exit..\n";
                terminal.selectionBeginIndex = terminal.text.length;
                terminal.selectionEndIndex = terminal.text.length;
            }
        ]]>
    </mx:Script>
    <mx:TextArea
        id="terminal"
        height="100%" width="100%"
        scroll="false"
        backgroundColor="0x003030"
        color="0xCCFF00"
        fontFamily="Lucida Console"
        fontSize="44"/>
</mx:WindowedApplication>
```

Im folgenden ActionScript-Beispiel für Flash wird ein einfaches Terminal im Vollbildmodus simuliert:

```
    import flash.display.Sprite;
    import flash.display.StageDisplayState;
    import flash.text.TextField;
    import flash.text.TextFormat;

    public class FullScreenTerminalExample extends Sprite
    {
        public function FullScreenTerminalExample():void
        {
            var terminal:TextField = new TextField();
            terminal.multiline = true;
            terminal.wordWrap = true;
            terminal.selectable = true;
            terminal.background = true;
            terminal.backgroundColor = 0x00333333;

            this.stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;

            addChild(terminal);
            terminal.width = 550;
            terminal.height = 400;

            terminal.text = "Welcome to the dumb terminal application. Press the ESC key to
exit.\n_";

            var tf:TextFormat = new TextFormat();
            tf.font = "Courier New";
            tf.color = 0x00CCFF00;
            tf.size = 12;
            terminal.setTextFormat(tf);

            terminal.setSelection(terminal.text.length - 1, terminal.text.length);
        }
    }
```

Kapitel 53: Anzeigebildschirme in AIR

Adobe AIR 1.0 und höher

Mit der Adobe® AIR® Screen-Klasse können Sie auf Informationen zu den Anzeigebildschirmen zugreifen, die an einen Computer oder ein Gerät angeschlossen sind.

Verwandte Hilfethemen

flash.display.Screen

Grundlagen zu Anzeigebildschirmen in AIR

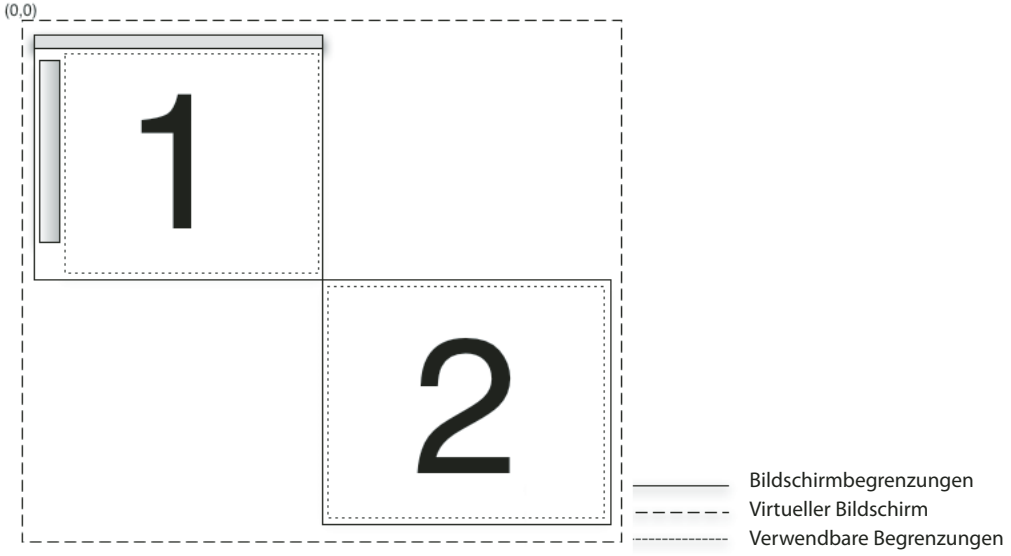
Adobe AIR 1.0 und höher

- Messen virtueller Desktops (Flex)
- Messen virtueller Desktops (Flash)

Die Screen-API enthält eine einzige Klasse, Screen, die statische Mitglieder zum Abrufen von Systembildschirminformationen und Instanzmitglieder zum Beschreiben eines bestimmten Bildschirms enthält.

An ein Computersystem können mehrere Monitore oder Anzeigen angeschlossen sein, die mehreren, in einem virtuellen Raum angeordneten Desktopbildschirmen entsprechen. Die AIR-Screen-Klasse bietet Informationen zu Bildschirmen, ihrer relativen Anordnung und dem nutzbaren Anzeigebereich. Wenn einem Bildschirm mehrere Monitore zugeordnet wurden, existiert nur ein Bildschirm. Ist der Bildschirm größer als der Anzeigebereich des Monitors, können Sie nicht ermitteln, welcher Bildschirmbereich derzeit angezeigt wird.

Ein Bildschirm repräsentiert einen unabhängigen Desktopanzeigebereich. Bildschirme werden innerhalb des virtuellen Desktops als Rechtecke beschrieben. Die obere linke Ecke des als primäre Anzeige festgelegten Bildschirms ist der Ursprung des virtuellen Desktopkoordinatensystems. Alle zum Beschreiben eines Bildschirms verwendeten Werte liegen in Pixeln vor.



In dieser Bildschirmanordnung gibt es auf dem virtuellen Desktop zwei Bildschirme. Die Koordinaten der oberen linken Ecke des Hauptbildschirms (1) sind immer (0,0). Wird die Bildschirmanordnung geändert und Bildschirm 2 als Hauptbildschirm festgelegt, nehmen die Koordinaten von Bildschirm 1 negative Werte an. Menüleisten, Taskleisten und Docks werden beim Angeben der Begrenzungen des verwendbaren Bereichs ignoriert.

Ausführliche Informationen über die Klassen, Methoden, Eigenschaften und Ereignisse der Bildschirm-API finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Aufzählen von Bildschirmen

Adobe AIR 1.0 und höher

Sie können die Bildschirme des virtuellen Desktops mit den folgenden screen-Methoden und -Eigenschaften aufzählen:

Methode oder Eigenschaft	Beschreibung
Screen.screens	Stellt ein Array von Screen-Objekten bereit, die die verfügbaren Screens beschreiben. Die Reihenfolge des Arrays spielt keine Rolle.
Screen.mainScreen	Stellt ein Screen-Objekt für den Hauptbildschirm bereit. Unter Mac OS X ist der Hauptbildschirm der Bildschirm, auf dem die Menüleiste angezeigt wird. Unter Windows ist der Hauptbildschirm der vom System festgelegte Primärbildschirm.
Screen.getScreensForRectangle()	Stellt ein Array von Screen-Objekten bereit, das die von einem gegebenen Rechteck geschnittenen Bildschirme beschreibt. Das an diese Methode übergebene Rechteck wird auf dem virtuellen Desktop in Pixelkoordinaten wiedergegeben. Wenn kein Bildschirm das Rechteck schneidet, ist das Array leer. Mit dieser Methode können Sie herausfinden, in welchen Bildschirmen ein Fenster angezeigt wird.

Speichern Sie die von den Methoden und Eigenschaften der Screen-Klasse zurückgegebenen Werte nicht. Die verfügbaren Bildschirme und deren Anordnung können jederzeit durch den Benutzer oder das Betriebssystem geändert werden.

Das folgende Beispiel verschiebt ein Fenster beim Betätigen der Pfeiltasten mithilfe der Bildschirm-API zwischen mehreren Bildschirmen. Um das Fenster zum nächsten Bildschirm zu verschieben, wird im Beispiel das `screens`-Array abgerufen und (je nach betätigter Pfeiltaste) vertikal oder horizontal sortiert. Der Code geht danach das sortierte Array durch und verlgeicht jeden Bildschirm mit den Koordinaten des aktuellen Bildschirms. Zum Identifizieren des aktuellen Bildschirms des Fensters ruft das Beispiel `Screen.getScreensForRectangle()` auf und übergibt die Fenstergrenzen.

```
package {
    import flash.display.Sprite;
    import flash.display.Screen;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;

    public class ScreenExample extends Sprite
    {
        public function ScreenExample()
        {
                stage.align = StageAlign.TOP_LEFT;
                stage.scaleMode = StageScaleMode.NO_SCALE;

                stage.addEventListener(KeyboardEvent.KEY_DOWN,onKey);
        }

        private function onKey(event:KeyboardEvent):void{
            if(Screen.screens.length > 1){
                switch(event.keyCode){
                    case Keyboard.LEFT :
                        moveLeft();
                        break;
                    case Keyboard.RIGHT :
                        moveRight();
                        break;
                    case Keyboard.UP :
                        moveUp();
                        break;
                    case Keyboard.DOWN :
                        moveDown();
                        break;
                }
            }
        }

        private function moveLeft():void{
            var currentScreen = getCurrentScreen();
            var left:Array = Screen.screens;
            left.sort(sortHorizontal);
            for(var i:int = 0; i < left.length - 1; i++){
                if(left[i].bounds.left < stage.nativeWindow.bounds.left){
                    stage.nativeWindow.x +=
                        left[i].bounds.left - currentScreen.bounds.left;
                    stage.nativeWindow.y += left[i].bounds.top - currentScreen.bounds.top;
                }
            }
        }
```

```
private function moveRight():void{
    var currentScreen:Screen = getCurrentScreen();
    var left:Array = Screen.screens;
    left.sort(sortHorizontal);
    for(var i:int = left.length - 1; i > 0; i--){
        if(left[i].bounds.left > stage.nativeWindow.bounds.left){
            stage.nativeWindow.x +=
                left[i].bounds.left - currentScreen.bounds.left;
            stage.nativeWindow.y += left[i].bounds.top - currentScreen.bounds.top;
        }
    }
}

private function moveUp():void{
    var currentScreen:Screen = getCurrentScreen();
    var top:Array = Screen.screens;
    top.sort(sortVertical);
    for(var i:int = 0; i < top.length - 1; i++){
        if(top[i].bounds.top < stage.nativeWindow.bounds.top){
            stage.nativeWindow.x += top[i].bounds.left - currentScreen.bounds.left;
            stage.nativeWindow.y += top[i].bounds.top - currentScreen.bounds.top;
            break;
        }
    }
}

private function moveDown():void{
    var currentScreen:Screen = getCurrentScreen();

    var top:Array = Screen.screens;
    top.sort(sortVertical);
    for(var i:int = top.length - 1; i > 0; i--){
        if(top[i].bounds.top > stage.nativeWindow.bounds.top){
            stage.nativeWindow.x += top[i].bounds.left - currentScreen.bounds.left;
            stage.nativeWindow.y += top[i].bounds.top - currentScreen.bounds.top;
            break;
        }
    }
}

private function sortHorizontal(a:Screen,b:Screen):int{
    if (a.bounds.left > b.bounds.left){
        return 1;
    } else if (a.bounds.left < b.bounds.left){
```

```
                return -1;
            } else {return 0;}
        }

        private function sortVertical(a:Screen,b:Screen):int{
            if (a.bounds.top > b.bounds.top){
                return 1;
            } else if (a.bounds.top < b.bounds.top){
                return -1;
            } else {return 0;}
        }

        private function getCurrentScreen():Screen{
            var current:Screen;
            var screens:Array = Screen.getScreensForRectangle(stage.nativeWindow.bounds);
            (screens.length > 0) ? current = screens[0] : current = Screen.mainScreen;
            return current;
        }
    }
}
```

Kapitel 54: Drucken

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Von Flash-Laufzeitumgebungen (wie Adobe® Flash® Player und Adobe® AIR™) kann eine Verbindung mit der Druckschnittstelle eines Betriebssystems hergestellt werden, sodass Seiten an die Druckwarteschlange weitergeleitet werden können. Alle Seiten, die an die Druckerwarteschlange gesendet werden, können sichtbare, dynamische oder nicht auf dem Bildschirm angezeigte Inhalte umfassen, einschließlich Datenbankwerten und dynamischem Text. Außerdem enthalten die Eigenschaften der flash.printing.PrintJob-Klasse Werte auf Grundlage der Druckereinstellungen eines Benutzers, sodass Sie die Seiten entsprechend formatieren können.

In Folgenden werden Strategien zur Verwendung der Methoden und Eigenschaften der flash.printing.PrintJob-Klasse beschrieben, mit denen Druckaufträge erstellt, die Druckeinstellungen eines Benutzers ermittelt sowie Druckaufträge entsprechend dem Feedback der Flash-Laufzeit und dem Betriebssystem des Benutzers geändert werden können.

Grundlagen zum Drucken

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 3.0 können Sie mithilfe der PrintJob-Klasse einen Schnappschuss des Anzeigeinhalts erstellen und in einem Ausdruck auf Papier umwandeln. Auf gewisse Weise entspricht die Festlegung des Druckbilds der Festlegung der Bildschirmanzeige, da Sie die Position und Größe von Elementen festlegen, um das gewünschte Layout zu erstellen. Das Drucken unterscheidet sich jedoch in einigen Merkmalen vom Layout auf dem Bildschirm. Drucker verfügen beispielsweise über eine andere Auflösung als Computerbildschirme. Der Inhalt eines Computerbildschirms ist dynamisch und kann sich ändern, während der gedruckte Inhalt per Definition statisch ist. Beim Drucken müssen zudem die Einschränkungen der festen Seitengröße sowie die Möglichkeit des Drucks mehrerer Seiten berücksichtigt werden.

Obwohl diese Unterschiede offensichtlich erscheinen, müssen sie beim Einrichten des Drucks mit ActionScript beachtet werden. Genaue Druckergebnisse hängen von der Kombination der von Ihnen angegebenen Werte und der Eigenschaften des verwendeten Druckers ab. Die PrintJob-Klasse enthält Eigenschaften, mit denen Sie wichtige Merkmale des vom Benutzer verwendeten Druckers bestimmen können.

Wichtige Konzepte und Begriffe

In der folgenden Liste sind wichtige Begriffe im Zusammenhang mit Druckvorgängen aufgeführt:

Druckwarteschlange Ein Teil des Betriebssystems oder des Druckertreibers, in dem Seiten vor dem Drucken gespeichert und dann an den Drucker gesendet werden, wenn dieser zur Verfügung steht.

Seitenausrichtung Die Drehung des Druckinhalts relativ zum Papier, entweder horizontal (Querformat) oder vertikal (Hochformat).

Druckauftrag Die Seite oder die Seiten, aus denen sich ein Ausdruck zusammensetzt.

Drucken einer Seite

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie verwenden für Druckvorgänge eine Instanz der PrintJob-Klasse. Zum Drucken einer einfachen Seite über Flash Player oder AIR werden nacheinander die folgenden vier Anweisungen verwendet:

- `new PrintJob()`: Erstellt eine neue Instanz des Druckauftrags mit dem angegebenen Namen.
- `PrintJob.start()`: Startet den Druckvorgang im Betriebssystem, öffnet das Druckdialogfeld für den Benutzer und gibt die schreibgeschützten Eigenschaften des Druckauftrags an.
- `PrintJob.addPage()`: Enthält detaillierte Daten zum Inhalt des Druckauftrags, wie das Sprite-Objekt (und zugehörige untergeordnete Objekte), die Größe des Druckbereichs sowie die Angabe, ob das Bild als Vektorgrafik oder Bitmapbild gedruckt werden soll. Mit aufeinander folgenden Aufrufen von `addPage()` können mehrere Sprites auf mehreren Seiten gedruckt werden.
- `PrintJob.send()`: Sendet die Seiten an den Drucker des Betriebssystems.

Im folgenden Beispiel wird ein einfaches Skript für einen Druckauftrag gezeigt (einschließlich der Anweisungen `package`, `import` und `class` für die Kompilierung):

```
package
{
    import flash.printing.PrintJob;
    import flash.display.Sprite;

    public class BasicPrintExample extends Sprite
    {
        var myPrintJob:PrintJob = new PrintJob();
        var mySprite:Sprite = new Sprite();

        public function BasicPrintExample()
        {
            myPrintJob.start();
            myPrintJob.addPage(mySprite);
            myPrintJob.send();
        }
    }
}
```

Hinweis: *In diesem Beispiel sind die Grundelemente eines Skripts für einen Druckauftrag dargestellt. Es enthält jedoch keine Fehlerverarbeitung. Informationen zum Erstellen eines Skripts, bei dem der Abbruch eines Druckauftrags durch den Benutzer korrekt verarbeitet wird, finden Sie unter „Ausnahmen und Rückgabewerte“ auf Seite 987.*

Wenn Sie die Eigenschaften eines PrintJob-Objekts aus einem bestimmten Grund entfernen möchten, setzen Sie die PrintJob-Variable auf `null` (z. B. `myPrintJob = null`).

Aufgaben in der Flash-Laufzeitumgebung und Drucken im System

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Da in Flash-Laufzeitumgebungen Seiten an die Druckschnittstelle des Betriebssystems weitergeleitet werden, sollten Sie den Umfang der von Flash-Laufzeitumgebungen sowie der von der Druckschnittstelle des Betriebssystems verwalteten Aufgaben kennen. In Flash-Laufzeitumgebungen können Druckaufträge eingeleitet, einige Seiteneinstellungen des Druckers gelesen sowie Inhalte für Druckaufträge an das Betriebssystem weitergeleitet werden. Darüber hinaus kann überprüft werden, ob ein Druckauftrag vom Benutzer oder vom System abgebrochen wurde. Andere Prozesse, z. B. Anzeigen druckerspezifischer Dialogfelder, Abbrechen eines Druckauftrags in der Druckwarteschlange oder Anzeigen des Druckerstatus, werden über das Betriebssystem durchgeführt. Flash-Laufzeitumgebungen können Fehler beim Initiieren oder Formatieren eines Druckauftrags beheben. Es können jedoch nur bestimmte Eigenschaften oder Fehlerzustände der Druckschnittstelle des Betriebssystems gemeldet werden. In dem von Entwicklern geschriebenen Code muss auf diese Eigenschaften oder Fehler entsprechend reagiert werden.

Ausnahmen und Rückgabewerte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Falls der Benutzer den Druckauftrag abgebrochen hat, sollten Sie zunächst überprüfen, ob die `PrintJob.start()`-Methode den Wert `true` zurückgibt, und erst dann `addPage()` und `send()` aufrufen. Sie können auf einfache Weise überprüfen, ob diese Methoden abgebrochen wurden, indem Sie sie wie folgt in eine `if`-Anweisung einschließen:

```
if (myPrintJob.start())
{
    // addPage() and send() statements here
}
```

Wenn `PrintJob.start()` auf `true` eingestellt ist, hat der Benutzer den Druckbefehl ausgewählt (oder eine Flash-Laufzeitumgebung, wie Flash Player oder AIR, hat einen Druckbefehl aufgerufen). Deshalb können die Methoden `addPage()` und `send()` aufgerufen werden.

Zur besseren Verwaltung des Druckvorgangs werden in Flash-Laufzeitumgebungen Ausnahmen für die `PrintJob.addPage()`-Methode ausgelöst, sodass Fehler abgefangen und dem Benutzer Informationen und Optionen bereitgestellt werden können. Wenn bei einer `PrintJob.addPage()`-Methode Fehler auftreten, können Sie zudem eine andere Funktion aufrufen oder den aktuellen Druckauftrag abbrechen. Sie können diese Ausnahmen abfangen, indem Sie `addPage()`-Aufrufe in eine `try..catch`-Anweisung einbetten, wie im folgenden Beispiel gezeigt. In diesem Beispiel ist `[params]` ein Platzhalter für die Parameter, mit denen der zu druckende Inhalt angegeben wird:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error,
    }
    myPrintJob.send();
}
```

Nachdem der Druckauftrag gestartet wurde, können Sie mit `PrintJob.addPage()` Inhalte hinzufügen und prüfen, ob dadurch eine Ausnahme ausgelöst wird (z. B. wenn der Benutzer den Druckauftrag abgebrochen hat). Wenn eine Ausnahme ausgelöst wird, können Sie Code zur `catch`-Anweisung hinzufügen, um dem Benutzer (oder der Flash-Laufzeitumgebung) Informationen oder Optionen bereitzustellen, oder den aktuellen Druckauftrag abbrechen. Wenn die Seite erfolgreich hinzugefügt wurde, können Sie fortfahren und die Seiten mit `PrintJob.send()` an den Drucker senden.

Wenn in der Flash-Laufzeitumgebung beim Senden des Druckauftrags an den Drucker ein Fehler auftritt (wenn der Drucker z. B. offline ist), können Sie auch diese Ausnahme abfangen und Informationen oder weitere Optionen bereitstellen (z. B. Anzeigen eines Meldungstexts oder einer Warnmeldung in der Animation). Sie können z. B. einem Textfeld neuen Text in einer `if..else`-Anweisung zuweisen, wie im folgenden Beispielcode gezeigt:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error.
    }
    myPrintJob.send();
}
else
{
    myAlert.text = "Print job canceled";
}
```

Ein praktisches Beispiel finden Sie unter „Druckbeispiel: Skalieren, Zuschneiden und Anpassen“ auf Seite 996.

Seiteneigenschaften

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Nachdem der Benutzer im Druckdialogfeld auf „OK“ geklickt hat und für `PrintJob.start()` der Wert `true` zurückgegeben wurde, können Sie auf die Eigenschaften zugreifen, die durch die Druckereinstellungen festgelegt sind. Zu diesen Einstellungen zählen die Papierhöhe und -breite (`pageHeight` und `pageWidth`) sowie die Ausrichtung des Inhalts auf dem Papier. Da es sich um Druckereinstellungen handelt, die nicht über die Flash-Laufzeitumgebung gesteuert werden, können Sie diese Einstellungen nicht ändern. Sie können jedoch die Inhalte, die an den Drucker gesendet werden, entsprechend den aktuellen Einstellungen ändern. Weitere Informationen finden Sie unter „Einrichten von Größe, Skalierung und Ausrichtung“ auf Seite 990.

Vektor- oder Bitmapdarstellung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können den Druckauftrag manuell so einrichten, dass jede Seite in der Druckwarteschlange als Vektorgrafik oder Bitmapbild verarbeitet wird. In einigen Fällen wird beim Vektordruck eine kleinere Datei in der Druckwarteschlange erstellt und ein besseres Bild erzielt als beim Bitmapdruck. Wenn der Inhalt jedoch ein Bitmapbild enthält und Sie die Alphatransparenz oder die Farbeffekte beibehalten möchten, drucken Sie die Seite als Bitmapbild. Bei einem nicht PostScript-fähigen Drucker werden alle Vektorgrafiken automatisch in Bitmapbilder umgewandelt.

Sie geben den Bitmapdruck an, indem Sie ein PrintJobOptions-Objekt als dritten Parameter von `PrintJob.addPage()` übergeben.

Für Flash Player und AIR-Versionen vor AIR 2 stellen Sie den `printAsBitmap`-Parameter des PrintJobOptions-Objekts auf `true` ein, wie im Folgenden gezeigt:

```
var options:PrintJobOptions = new PrintJobOptions();
options.printAsBitmap = true;
myPrintJob.addPage(mySprite, null, options);
```

Wenn Sie keinen Wert für den dritten Parameter angeben, wird der Druckauftrag mit der Standardoption gedruckt, d. h. als Vektordruck.

Für AIR 2 und spätere Versionen verwenden Sie die `printMethod`-Eigenschaft des PrintJobOptions-Objekts, um die Druckmethode anzugeben. Diese Eigenschaft akzeptiert drei Werte, die als Konstanten in der PrintMethod-Klasse definiert sind:

- `PrintMethod.AUTO`: Wählt automatisch die beste Druckmethode auf Grundlage des zu druckenden Inhalts. Wenn eine Seite beispielsweise Text enthält, wird der Vektordruck gewählt. Wenn jedoch ein Wasserzeichen mit Alphatransparenz über dem Text gedruckt wird, wird der Bitmapdruck gewählt, um die Transparenz beizubehalten.
- `PrintMethod.BITMAP`: Erzwingt den Bitmapdruck unabhängig vom Inhalt.
- `PrintMethod.VECTOR`: Erzwingt den Vektordruck unabhängig vom Inhalt.

Zeitbeschränkung bei Anweisungen für Druckaufträge

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In ActionScript 3.0 ist ein PrintJob-Objekt nicht auf ein einzelnes Bild beschränkt (wie dies in früheren Versionen von ActionScript der Fall war). Da im Betriebssystem jedoch Informationen zum Druckstatus angezeigt werden, nachdem der Benutzer im Druckdialogfeld auf die Schaltfläche „OK“ geklickt hat, sollten Sie möglichst bald `PrintJob.addPage()` und `PrintJob.send()` aufrufen, damit Seiten an die Druckwarteschlange gesendet werden. Durch eine Verzögerung im Bild mit dem `PrintJob.send()`-Aufruf wird auch der Druckvorgang verzögert.

In ActionScript 3.0 gilt ein Skriptzeitlimit von 15 Sekunden. Zwischen den einzelnen Hauptanweisungen in einem Druckauftrag dürfen daher 15 Sekunden nicht überschritten werden. Mit anderen Worten, das Skriptzeitlimit von 15 Sekunden gilt für folgende Zeitintervalle:

- zwischen `PrintJob.start()` und der ersten `PrintJob.addPage()`-Anweisung
- zwischen `PrintJob.addPage()` und der nächsten `PrintJob.addPage()`-Anweisung
- zwischen der letzten `PrintJob.addPage()`-Anweisung und `PrintJob.send()`

Beim Überschreiten des Limits von 15 Sekunden bei einem dieser Intervalle wird beim nächsten Aufruf von `PrintJob.start()` für die PrintJob-Instanz der Wert `false` zurückgegeben und beim nächsten Aufruf von `PrintJob.addPage()` für die PrintJob-Instanz in Flash Player oder AIR eine Laufzeitausnahme ausgelöst.

Einrichten von Größe, Skalierung und Ausrichtung

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Im Abschnitt „Drucken einer Seite“ auf Seite 986 werden ausführlich die Schritte für einen einfachen Druckauftrag beschrieben, bei dem die Druckausgabe genau der Bildschirmgröße und -position des angegebenen Sprite-Objekts entspricht. Bei verschiedenen Druckern kommen jedoch unterschiedliche Druckauflösungen zum Einsatz. Zudem können Druckeinstellungen festgelegt sein, die sich nachteilig auf die Darstellung des gedruckten Sprite-Objekts auswirken.

In Flash-Laufzeitumgebungen können die Druckeinstellungen des Betriebssystems gelesen werden. Es handelt sich dabei jedoch um schreibgeschützte Eigenschaften: Sie können diese Werte zwar anzeigen, jedoch nicht ändern. Sie können beispielsweise die Einstellung des Druckers für die Seitengröße ermitteln und den Inhalt dann so anpassen, dass er dieser Größe entspricht. Sie können auch die Einstellungen für die Seitenränder und die Seitenausrichtung eines Druckers ermitteln. Um den Inhalt entsprechend den Druckereinstellungen anzupassen, sollten Sie einen Druckbereich festlegen, die Unterschiede zwischen der Bildschirmauflösung und den Punktmaßen des Druckers abgleichen oder den Inhalt so ändern, dass er den Einstellungen für die Seitengröße und die Ausrichtung des Druckers entspricht.

Verwenden von Rectangle-Objekten für den Druckbereich

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `PrintJob.addPage()`-Methode können Sie den Druckbereich für ein Sprite-Objekt festlegen. Der zweite Parameter `printArea` hat die Struktur eines Rectangle-Objekts. Sie haben drei Möglichkeiten, den Wert für diesen Parameter anzugeben:

- Sie können ein Rectangle-Objekt mit bestimmten Eigenschaften erstellen und dieses Objekt dann im `addPage()`-Aufruf verwenden, wie im folgenden Beispiel dargestellt:

  ```
  private var rect1:Rectangle = new Rectangle(0, 0, 400, 200);
  myPrintJob.addPage(sheet, rect1);
  ```

- Wenn Sie zuvor kein Rectangle-Objekt angegeben haben, können Sie es im Aufruf direkt angeben, wie im folgenden Beispiel dargestellt:

  ```
  myPrintJob.addPage(sheet, new Rectangle(0, 0, 100, 100));
  ```

- Wenn Sie Werte für den dritten Parameter im `addPage()`-Aufruf angeben, jedoch kein Rectangle-Objekt festlegen möchten, können Sie für den zweiten Parameter `null` angeben, wie im folgenden Beispiel dargestellt:

  ```
  myPrintJob.addPage(sheet, null, options);
  ```

Punkt und Pixel im Vergleich

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Breite und Höhe eines Rechtecks werden in Pixel angegeben. Bei Druckern wird die Druckmaßeinheit Punkt verwendet. Punkt ist eine feste Größe (1/72 Zoll), die Größe eines Pixels richtet sich jedoch nach der jeweiligen Auflösung des Bildschirms. Der Umrechnungsfaktor zwischen Pixel und Punkt hängt daher von den Druckereinstellungen und davon ab, ob das Sprite skaliert ist. Ein nicht skalierter Sprite mit einer Breite von 72 Pixel wird mit einer Breite von 1 Zoll gedruckt. Dabei entspricht ein Punkt einem Pixel, unabhängig von der Bildschirmauflösung.

Zoll- oder Zentimeterangaben können folgendermaßen in Twip oder Punkt umgerechnet werden (ein Twip ist 1/20 Punkt):

- 1 Punkt = 1/72 Zoll = 20 Twip
- 1 Zoll = 72 Punkt = 1440 Twip
- 1 cm = 567 Twip

Wenn Sie den Parameter `printArea` weglassen oder falsch übergeben, wird der gesamte Sprite-Bereich gedruckt.

Skalieren

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn Sie ein Sprite-Objekt vor dem Drucken skalieren möchten, legen Sie die Skalierungseigenschaften (siehe „Ändern der Größe und Skalieren von Objekten“ auf Seite 191) vor dem Aufrufen der `PrintJob.addPage()`-Methode fest und setzen Sie sie nach dem Drucken wieder auf die ursprünglichen Werte zurück. Die Skalierung eines Sprite-Objekts hat keinen Bezug zur `printArea`-Eigenschaft. Wenn Sie beispielsweise einen Druckbereich mit der Größe 50 x 50 Pixel festlegen, werden 2500 Pixel gedruckt. Wenn Sie das Sprite-Objekt skalieren, werden dieselben 2500 Pixel in der skalierten Größe gedruckt.

Ein Beispiel finden Sie unter „Druckbeispiel: Skalieren, Zuschneiden und Anpassen“ auf Seite 996.

Drucken im Querformat oder Hochformat

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Da es in Flash Player und AIR möglich ist, die Einstellungen für die Druckausrichtung zu ermitteln, können Sie ActionScript-Code programmieren, mit dem die Größe oder Drehung des Inhalts entsprechend den Druckereinstellungen angepasst werden, wie im folgenden Beispiel dargestellt:

```
if (myPrintJob.orientation == PrintJobOrientation.LANDSCAPE)
{
    mySprite.rotation = 90;
}
```

Hinweis: *Wenn Sie die Systemeinstellungen für die Ausrichtung des Inhalts auf dem Papier einlesen möchten, denken Sie daran, die PrintJobOrientation-Klasse zu importieren. Die PrintJobOrientation-Klasse enthält Konstanten, mit denen die Ausrichtung des Inhalts auf der Seite definiert wird. Sie importieren die Klasse mit der folgenden Anweisung:*

```
import flash.printing.PrintJobOrientation;
```

Anpassen der Seitenhöhe und Seitenbreite

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ähnlich wie bei den Druckereinstellungen für die Druckausrichtung können Sie die Einstellungen für die Seitenhöhe und die Seitenbreite ermitteln und entsprechend anpassen, indem Sie Code in eine `if`-Anweisung einbetten. Der folgende Code ist ein Beispiel hierfür:

```
if (mySprite.height > myPrintJob.pageHeight)
{
    mySprite.scaleY = .75;
}
```

Zusätzlich können die Einstellungen für die Seitenränder durch einen Vergleich der Seiten- und Papierabmessungen ermittelt werden, wie im folgenden Beispiel dargestellt:

```
margin_height = (myPrintJob.paperHeight - myPrintJob.pageHeight) / 2;
margin_width = (myPrintJob.paperWidth - myPrintJob.pageWidth) / 2;
```

Erweiterte Drucktechniken

Adobe AIR 2 und höher

Ab Adobe AIR 2 hat die PrintJob-Klasse zusätzliche Eigenschaften und Methoden. Außerdem werden drei neue Klassen unterstützt: PrintUIOptions, PaperSize und PrintMethod. Diese Änderungen ermöglichen zusätzliche Arbeitsabläufe beim Drucken und bieten Autoren eine bessere Steuerung des Druckvorgangs. Zu den Änderungen zählen:

- Dialogfelder für die Seiteneinrichtung: Sowohl standardmäßige als auch benutzerdefinierte Dialogfelder für die Seiteneinrichtung können angezeigt werden. Der Benutzer kann vor dem Drucken die Seitenbereiche, das Papierformat, die Ausrichtung und die Skalierung festlegen.
- Druckansicht: Sie können einen Ansichtsmodus erstellen, der Papierformat, Ränder und die Position des Inhalts auf der Seite genau zeigt.
- Eingeschränktes Drucken: Autoren können die Druckoptionen einschränken, wie beispielsweise den druckbaren Seitenbereich.
- Qualitätsoptionen: Autoren können die Druckqualität für ein Dokument anpassen und Benutzern die Möglichkeit geben, die Auflösung und Farboptionen auszuwählen.
- Mehrere Drucksitzungen: Eine einzelne PrintJob-Instanz kann nun für mehrere Drucksitzungen verwendet werden. Anwendungen können bei jeder Anzeige der Dialogfelder „Seite einrichten“ und „Drucken“ einheitliche Einstellungen bereitstellen.

Änderungen am Druckablauf

Der neue Arbeitsablauf beim Drucken setzt sich aus den folgenden Schritten zusammen:

- `new PrintJob()`: Erstellt eine PrintJob-Instanz (oder verwendet eine vorhandene Instanz erneut). Zahlreiche neue PrintJob-Eigenschaften und -Methoden, wie beispielsweise `selectPaperSize()`, stehen vor dem Start des Druckauftrags oder während des Druckvorgangs zur Verfügung.
- `PrintJob.showPageSetupDialog()`: Zeigt das Dialogfeld „Seite einrichten“ an, ohne einen Druckauftrag zu starten (optional).
- `PrintJob.start()` oder `PrintJob.start2()`: Zusätzlich zur `start()`-Methode wird die `start2()`-Methode verwendet, um den Prozess für die Druckerwarteschlange einzuleiten. Mit der `start2()`-Methode können Sie festlegen, ob das Dialogfeld „Drucken“ angezeigt werden soll. Wenn ja, können Sie das Dialogfeld auch anpassen.
- `PrintJob.addPage()`: Fügt dem Druckauftrag Inhalt hinzu. Dies hat sich im Vergleich mit dem vorhandenen Prozess nicht geändert.
- `PrintJob.send()` oder `PrintJob.terminate()`: Sendet die Seiten an den ausgewählten Drucker oder beendet den Druckauftrag, ohne die Seiten zu senden. Druckaufträge werden als Reaktion auf einen Fehler beendet. Wenn eine PrintJob-Instanz beendet wurde, kann sie dennoch wieder verwendet werden. Bei der Wiederverwendung der PrintJob-Instanz werden die aktuellen Druckeinstellungen beibehalten, unabhängig davon, ob der Druckauftrag an den Drucker gesendet oder beendet wird.

Dialogfeld „Seite einrichten“

Die `showPageSetupDialog()`-Methode zeigt das Dialogfeld „Seite einrichten“ des Betriebssystems an, sofern von der aktuellen Umgebung unterstützt. Überprüfen Sie vor dem Aufruf dieser Methode immer die `supportsPageSetupDialog`-Eigenschaft. Ein einfaches Beispiel:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
//check for static property supportsPageSetupDialog of PrintJob class
if (PrintJob.supportsPageSetupDialog) {
    myPrintJob.showPageSetupDialog();
}
```

Die Methode kann wahlweise mit einer Eigenschaft der PrintUIOptions-Klasse aufgerufen werden, um zu steuern, welche Optionen im Dialogfeld „Seite einrichten“ angezeigt werden. Die minimale und maximale Seitenanzahl kann festgelegt werden. Mit dem folgenden Beispiel werden nur die ersten drei Seiten gedruckt:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
if (PrintJob.supportsPageSetupDialog) {
    var uiOpt:PrintUIOptions = new PrintUIOptions();
    uiOpt.minPage = 1;
    uiOpt.maxPage = 3;
    myPrintJob.showPageSetupDialog(uiOpt);
}
```

Ändern der Druckeinstellungen

Die Einstellungen für eine PrintJob-Instanz können jederzeit nach der Konstruktion der Instanz geändert werden. Die Einstellungen können zwischen `addPage()`-Aufrufen und nach dem Senden oder Beenden eines Druckauftrags geändert werden. Einige Einstellungen, wie beispielsweise die `printer`-Eigenschaft, gelten für den ganzen Druckauftrag, nicht nur für einzelne Seiten. Diese Einstellungen müssen vor einem Aufruf von `start()` oder `start2()` festgelegt werden.

Die `selectPaperSize()`-Methode kann aufgerufen werden, um das standardmäßige Papierformat in den Dialogfeldern „Seite einrichten“ und „Drucken“ festzulegen. Sie kann auch während eines Druckauftrags aufgerufen werden, um das Papierformat für einen Seitenbereich festzulegen. Sie wird mithilfe von Konstanten aufgerufen, die in der `PaperSize`-Klasse definiert sind, wie in diesem Beispiel, in dem ein Umschlag der Größe 10 ausgewählt wird:

```
import flash.printing.PrintJob;
import flash.printing.PaperSize;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.selectPaperSize(PaperSize.ENV_10);
```

Verwenden Sie die `printer`-Eigenschaft, um den Namen des Druckers für den aktuellen Druckauftrag abzurufen oder festzulegen. Standardmäßig ist sie auf den Namen des Standarddruckers eingestellt. Die `printer`-Eigenschaft hat den Wert `null`, wenn keine Drucker verfügbar sind oder wenn das Drucken vom System nicht unterstützt wird. Zum Ändern des Druckers rufen Sie zunächst die Liste der verfügbaren Drucker mithilfe der `printers`-Eigenschaft ab. Dies ist eine Vector-Eigenschaft, deren String-Elemente die verfügbaren Druckernamen sind. Stellen Sie die `printer`-Eigenschaft auf einen dieser Stringwerte ein, um den jeweiligen Drucker als aktiven Drucker festzulegen. Die `printer`-Eigenschaft eines aktiven Druckauftrags kann nicht geändert werden. Wenn Sie versuchen, die Eigenschaft nach einem erfolgreichen Aufruf von `start()` oder `start2()` und vor dem Senden oder Beenden des Druckauftrags zu ändern, schlägt der Vorgang fehl. Im folgenden Beispiel wird gezeigt, wie diese Eigenschaft festgelegt werden kann:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.printer = "HP_LaserJet_1";
myPrintJob.start();
```

Mit der `copies`-Eigenschaft wird der Wert für die Anzahl der Exemplare abgerufen, der im Dialogfeld „Drucken“ des Betriebssystems festgelegt ist. Mit den Eigenschaften `firstPage` und `lastPage` wird der Seitenbereich abgerufen. Mit der `orientation`-Eigenschaft wird die Einstellung für die Papierausrichtung abgerufen. Diese Eigenschaften können festgelegt werden, um die Werte im Dialogfeld „Drucken“ außer Kraft zu setzen. Im folgenden Beispiel werden diese Eigenschaften festgelegt:

```
import flash.printing.PrintJob;
import flash.printing.PrintJobOrientation;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.copies = 3;
myPrintJob.firstPage = 1;
myPrintJob.lastPage = 3;
myPrintJob.orientation = PrintJobOrientation.LANDSCAPE;
```

Die folgenden schreibgeschützten Einstellungen für `PrintJob` bieten nützliche Informationen zur aktuellen Druckereinrichtung:

- `paperArea`: Die rechteckigen Grenzen des Druckmediums in Punkten.
- `printableArea`: Die rechteckigen Grenzen des Druckbereichs in Punkten.
- `maxPixelsPerInch`: Die tatsächliche Auflösung des aktuellen Druckers in Pixel pro Zoll.
- `isColor`: Gibt an, ob der aktuelle Drucker den Farbdruck unterstützt (gibt `true` zurück, wenn der aktuelle Drucker den Farbdruck unterstützt).

Siehe „Druckbeispiel: Seiteneinrichtung und Druckoptionen“ auf Seite 998.

Druckbeispiel: Mehrseitiger Druck

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Drucken mehrerer Inhaltsseiten können Sie die einzelnen Inhaltsseiten jeweils mit einem unterschiedlichen Sprite verknüpfen (in diesem Fall `sheet1` und `sheet2`) und dann für jeden Sprite `PrintJob.addPage()` verwenden. Diese Vorgehensweise ist im folgenden Codebeispiel dargestellt:

```
package
{
    import flash.display.MovieClip;
    import flash.printing.PrintJob;
    import flash.printing.PrintJobOrientation;
    import flash.display.Stage;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.geom.Rectangle;

    public class PrintMultiplePages extends MovieClip
    {
        private var sheet1:Sprite;
        private var sheet2:Sprite;

        public function PrintMultiplePages():void
        {
            init();
            printPages();
        }

        private function init():void
        {
            sheet1 = new Sprite();
            createSheet(sheet1, "Once upon a time...", {x:10, y:50, width:80, height:130});
            sheet2 = new Sprite();
            createSheet(sheet2, "There was a great story to tell, and it ended quickly.\n\nThe
end.", null);
        }

        private function createSheet(sheet:Sprite, str:String, imgValue:Object):void
        {
            sheet.graphics.beginFill(0xEEEEEE);
            sheet.graphics.lineStyle(1, 0x000000);
            sheet.graphics.drawRect(0, 0, 100, 200);
            sheet.graphics.endFill();

            var txt:TextField = new TextField();
            txt.height = 200;
            txt.width = 100;
            txt.wordWrap = true;
            txt.text = str;

            if (imgValue != null)
            {
                var img:Sprite = new Sprite();
                img.graphics.beginFill(0xFFFFFF);
                img.graphics.drawRect(imgValue.x, imgValue.y, imgValue.width, imgValue.height);
                img.graphics.endFill();
                sheet.addChild(img);
            }
            sheet.addChild(txt);
        }

        private function printPages():void
        {
            var pj:PrintJob = new PrintJob();
```

```
            var pagesToPrint:uint = 0;
            if (pj.start())
            {
                if (pj.orientation == PrintJobOrientation.LANDSCAPE)
                {
                    throw new Error("Page is not set to an orientation of portrait.");
                }

                sheet1.height = pj.pageHeight;
                sheet1.width = pj.pageWidth;
                sheet2.height = pj.pageHeight;
                sheet2.width = pj.pageWidth;

                try
                {
                    pj.addPage(sheet1);
                    pagesToPrint++;
                }
                catch (error:Error)
                {
                // Respond to error.
                }

                try
                {
                    pj.addPage(sheet2);
                    pagesToPrint++;
                }
                catch (error:Error)
                {
                    // Respond to error.
                }

                if (pagesToPrint > 0)
                {
                    pj.send();
                }
            }
        }
    }
}
```

Druckbeispiel: Skalieren, Zuschneiden und Anpassen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei Bedarf können Sie die Größe (oder andere Eigenschaften) eines Anzeigeobjekts beim Drucken anpassen, um so Unterschiede zwischen der Darstellung auf dem Bildschirm und der Druckversion auf Papier auszugleichen. Beim Anpassen der Eigenschaften eines Anzeigeobjekts vor dem Drucken (z. B. mit den Eigenschaften `scaleX` und `scaleY`) müssen Sie beachten, dass das Objekt zugeschnitten wird, wenn es größer als das für den Druckbereich festgelegte Rectangle-Objekt ist. Es empfiehlt sich außerdem, die geänderten Eigenschaften nach dem Drucken der Seiten wieder zurückzusetzen.

Mit dem folgenden Code werden die Abmessungen des `txt`-Anzeigeobjekts skaliert (nicht jedoch der grüne Feldhintergrund). Das Textfeld wird entsprechend den Abmessungen des angegebenen Rectangle-Objekts zugeschnitten. Nach dem Drucken wird das Textfeld für die Anzeige auf dem Bildschirm auf die ursprüngliche Größe zurückgesetzt. Wenn der Benutzer den Druckauftrag im Druckdialogfeld des Betriebssystems abbricht, ändert sich der in der Flash-Laufzeitumgebung angezeigte Inhalt, um den Benutzer darüber zu informieren, dass der Druckauftrag abgebrochen wurde.

```
package
{
    import flash.printing.PrintJob;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.display.Stage;
    import flash.geom.Rectangle;

    public class PrintScaleExample extends Sprite
    {
        private var bg:Sprite;
        private var txt:TextField;

        public function PrintScaleExample():void
        {
            init();
            draw();
            printPage();
        }

        private function printPage():void
        {
            var pj:PrintJob = new PrintJob();
            txt.scaleX = 3;
            txt.scaleY = 2;
            if (pj.start())
            {
                trace(">> pj.orientation: " + pj.orientation);
                trace(">> pj.pageWidth: " + pj.pageWidth);
                trace(">> pj.pageHeight: " + pj.pageHeight);
                trace(">> pj.paperWidth: " + pj.paperWidth);
                trace(">> pj.paperHeight: " + pj.paperHeight);

                try
                {
                    pj.addPage(this, new Rectangle(0, 0, 100, 100));
                }
                catch (error:Error)
                {
                    // Do nothing.
                }
                pj.send();
            }
            else
            {
                txt.text = "Print job canceled";
            }
            // Reset the txt scale properties.
            txt.scaleX = 1;
```

```
            txt.scaleY = 1;
        }

        private function init():void
        {
            bg = new Sprite();
            bg.graphics.beginFill(0x00FF00);
            bg.graphics.drawRect(0, 0, 100, 200);
            bg.graphics.endFill();

            txt = new TextField();
            txt.border = true;
            txt.text = "Hello World";
        }

        private function draw():void
        {
            addChild(bg);
            addChild(txt);
            txt.x = 50;
            txt.y = 50;
        }
    }
}
```

Druckbeispiel: Seiteneinrichtung und Druckoptionen

Adobe AIR 2 und höher

Mit dem folgenden Beispiel werden die PrintJob-Einstellungen für Anzahl der Exemplare, Papierformat (Legal) und Seitenausrichtung (Querformat) initialisiert. Zuerst wird das Dialogfeld „Seite einrichten“ angezeigt, dann beginnt der Druckauftrag, indem das Dialogfeld „Drucken“ angezeigt wird.

```
package
{
    import flash.printing.PrintJob;
    import flash.printing.PrintJobOrientation;
    import flash.printing.PaperSize;
    import flash.printing.PrintUIOptions;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.display.Stage;
    import flash.geom.Rectangle;

    public class PrintAdvancedExample extends Sprite
    {
        private var bg:Sprite = new Sprite();
        private var txt:TextField = new TextField();
        private var pj:PrintJob = new PrintJob();
        private var uiOpt:PrintUIOptions = new PrintUIOptions();

        public function PrintAdvancedExample():void
        {
            initPrintJob();
```

```
        initContent();
        draw();
        printPage();
    }

    private function printPage():void
    {
        //test for dialog support as a static property of PrintJob class
        if (PrintJob.supportsPageSetupDialog)
        {
            pj.showPageSetupDialog();
        }
        if (pj.start2(uiOpt, true))
        {
            try
            {
                pj.addPage(this, new Rectangle(0, 0, 100, 100));
            }
            catch (error:Error)
            {
                // Do nothing.
            }
            pj.send();
        }
        else
        {
            txt.text = "Print job terminated";
            pj.terminate();
        }
    }

    private function initContent():void
    {
        bg.graphics.beginFill(0x00FF00);
        bg.graphics.drawRect(0, 0, 100, 200);
        bg.graphics.endFill();

        txt.border = true;
```

```
            txt.text = "Hello World";
        }

        private function initPrintJob():void
        {
            pj.selectPaperSize(PaperSize.LEGAL);
            pj.orientation = PrintJobOrientation.LANDSCAPE;
            pj.copies = 2;
            pj.jobName = "Flash test print";
        }

        private function draw():void
        {
            addChild(bg);
            addChild(txt);
            txt.x = 50;
            txt.y = 50;
        }
    }
}
```

Kapitel 55: Geolokation

Wenn ein Gerät Ortungsfunktionen unterstützt, können Sie die Geolokation-API verwenden, um die aktuelle geografische Position des Geräts abzurufen. Wenn ein Gerät dieses Funktionsmerkmal unterstützt, können Sie Informationen zur Geolokation abrufen. Zu diesen Informationen zählen Höhe, Genauigkeit, Richtung, Geschwindigkeit und Zeitstempel der letzten Positionsänderung.

Die Geolocation-Klasse löst `update`-Ereignisse als Reaktion auf den Positionssensor des Geräts aus. Das `update`-Ereignis ist ein GeolocationEvent-Objekt.

Verwandte Hilfethemen

flash.sensors.Geolocation

flash.events.GeolocationEvent

Erkennen von Geolokationsänderungen

Zur Verwendung des Sensors für die Geolokation instanziieren Sie ein Geolocation-Objekt und registrieren Sie die von diesem Objekt ausgelösten `update`-Ereignisse. Das `update`-Ereignis ist ein Geolocation-Ereignisobjekt. Das Ereignis hat acht Eigenschaften:

- `altitude` – Die Höhe in Metern.
- `heading` – Die Bewegungsrichtung in Grad (relativ zum geografischen Norden).
- `horizontalAccuracy` – Die horizontale Genauigkeit in Metern.
- `latitude` – Der Breitengrad.
- `longitude` – Der Längengrad.
- `speed` – Die Geschwindigkeit in Metern pro Sekunde.
- `timestamp` – Die Anzahl der Millisekunden seit der Initialisierung der Laufzeitumgebung bis zum Auftreten des Ereignisses.
- `verticalAccuracy` – Die vertikale Genauigkeit in Metern.

Bei der `timestamp`-Eigenschaft handelt es sich um ein Objekt mit dem int-Datentyp. Alle anderen Eigenschaften sind Number-Objekte.

Im Folgenden sehen Sie ein einfaches Beispiel zur Anzeige von Geolokationsdaten in einem Textfeld:

```
var geo:Geolocation;
if (Geolocation.isSupported)
{
    geo = new Geolocation();
    geo.addEventListener(GeolocationEvent.UPDATE, updateHandler);
}
else
{
    geoTextField.text = "Geolocation feature not supported";
}
function updateHandler(event:GeolocationEvent):void
{
    geoTextField.text = "latitude: " + event.latitude.toString() + "\n"
            + "longitude: " + event.longitude.toString() + "\n"
            + "altitude: " + event.altitude.toString()
            + "speed: " + event.speed.toString()
            + "heading: " + event.heading.toString()
            + "horizontal accuracy: " + event.horizontalAccuracy.toString()
            + "vertical accuracy: " + event.verticalAccuracy.toString()
}
```

Bevor Sie diesen Code verwenden können, müssen Sie das `geoTextField`-Textfeld erstellen und der Anzeigeliste hinzufügen.

Sie können das gewünschte Zeitintervall für Geolocation-Ereignisse anpassen, indem Sie die `setRequestedUpdateInterval()`-Methode für das Geolocation-Objekt aufrufen. Diese Methode akzeptiert einen Parameter, `interval`. Dies ist das angeforderte Aktualisierungsintervall in Millisekunden:

```
var geo:Geolocation = new Geolocation();
geo.setRequestedUpdateInterval(10000);
```

Die tatsächliche Zeitspanne zwischen Geolokationsaktualisierungen kann größer oder kleiner als dieser Wert sein. Änderungen am Aktualisierungsintervall betreffen alle registrierten Listener. Wenn Sie die `setRequestedUpdateInterval()`-Methode nicht aufrufen, werden die Aktualisierungen in der Anwendung gemäß dem Standardintervall des Geräts durchgeführt.

Der Benutzer kann verhindern, dass eine Anwendung auf Geolokationsdaten zugreift. Beispielsweise wird der Benutzer auf dem iPhone zur Bestätigung aufgefordert, wenn eine Anwendung versucht, Geolokationsdaten abzurufen. Der Benutzer hat dann die Möglichkeit, den Anwendungszugriff auf die Geolokationsdaten zu verweigern. Das Geolocation-Objekt löst ein `status`-Ereignis aus, wenn der Benutzer den Zugriff auf Geolokationsdaten verhindert. Weiterhin verfügt das Geolocation-Objekt über die `muted`-Eigenschaft, die auf `true` gesetzt ist, wenn der Sensor für die Geolokation nicht verfügbar ist. Das Geolocation-Objekt löst ein `status`-Ereignis aus, wenn die `muted`-Eigenschaft sich ändert. Der folgende Code zeigt, wie festgestellt werden kann, dass Geolokationsdaten nicht zur Verfügung stehen:

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.GeolocationEvent;
    import flash.events.MouseEvent;
    import flash.events.StatusEvent;
    import flash.sensors.Geolocation;
    import flash.text.TextField;
    import flash.text.TextFormat;

    public class GeolocationTest extends Sprite
    {

        private var geo:Geolocation;
        private var log:TextField;

        public function GeolocationTest()
        {
            super();
            stage.align = StageAlign.TOP_LEFT;
            stage.scaleMode = StageScaleMode.NO_SCALE;
            setUpTextField();

            if (Geolocation.isSupported)
            {
                geo = new Geolocation();
                if (!geo.muted)
                {
                    geo.addEventListener(GeolocationEvent.UPDATE, geoUpdateHandler);
                }
                geo.addEventListener(StatusEvent.STATUS, geoStatusHandler);
            }
            else
            {
                log.text = "Geolocation not supported";
            }
        }

        public function geoUpdateHandler(event:GeolocationEvent):void
        {
            log.text = "latitude : " + event.latitude.toString() + "\n";
            log.appendText("longitude : " + event.longitude.toString() + "\n");
        }

        public function geoStatusHandler(event:StatusEvent):void
        {
            if (geo.muted)
                geo.removeEventListener(GeolocationEvent.UPDATE, geoUpdateHandler);
            else
                geo.addEventListener(GeolocationEvent.UPDATE, geoStatusHandler);
        }

        private function setUpTextField():void
```

```
        {
            log = new TextField();
            var format:TextFormat = new TextFormat("_sans", 24);
            log.defaultTextFormat = format;
            log.border = true;
            log.wordWrap = true;
            log.multiline = true;
            log.x = 10;
            log.y = 10;
            log.height = stage.stageHeight - 20;
            log.width = stage.stageWidth - 20;
            log.addEventListener(MouseEvent.CLICK, clearLog);
            addChild(log);
        }
        private function clearLog(event:MouseEvent):void
        {
            log.text = "";
        }
    }
}
```

Hinweis: *iPhones der ersten Generation, die nicht mit einer GPS-Einheit ausgestattet sind, lösen nur gelegentlich* `update`*-Ereignisse aus. Bei diesen Geräten setzt ein Geolocation-Objekt anfänglich ein oder zwei* `update`*-Ereignisse ab. Danach werden* `update`*-Ereignisse abgesetzt, wenn sich Informationen deutlich ändern.*

Überprüfen der Unterstützung für die Geolokation

Mithilfe der `Geolocation.isSupported`-Eigenschaft können Sie überprüfen, ob die Laufzeitumgebung dieses Funktionsmerkmal unterstützt:

```
if (Geolocation.isSupported)
{
    // Set up geolocation event listeners and code.
}
```

Derzeit wird die Geolokation nur in ActionScript-Anwendungen für das iPhone und in Flash Lite 4 unterstützt. Wenn `Geolocation.isSupported` zur Laufzeit `true` lautet, wird die Geolokation unterstützt.

Einige iPhone-Modelle sind nicht mit einer GPS-Einheit ausgestattet. Bei diesen Modellen werden Geolokationsdaten mit anderen Methoden abgerufen (wie beispielsweise Triangulation über das Mobiltelefon). Bei diesen Modellen und bei einem iPhone, auf dem GPS deaktiviert ist, kann ein Geolocation-Objekt nur ein oder zwei anfängliche `update`-Ereignisse auslösen.

Kapitel 56: Internationalisierung von Anwendungen

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Mit dem flash.globalization-Paket lassen sich problemlos internationale Softwareprogramme erstellen, die an verschiedene Sprachen und geografische Regionen angepasst werden können.

Verwandte Hilfethemen

flash.globalization-Paket

„Lokalisieren von Anwendungen“ auf Seite 1023

Charles Bihis: Want to Localize your Flex/AIR Apps?

Internationalisierung von Anwendungen – Grundlagen

Die Begriffe „Globalisierung“ und „Internationalisierung“ haben im allgemeinen Sprachgebrauch häufig dieselbe Bedeutung. Doch laut den meisten Definitionen bezieht sich „Globalisierung“ auf eine Kombination aus geschäftlichen und technischen Prozessen, während „Internationalisierung“ ausschließlich technische Prozesse betrifft.

Im Folgenden werden einige wichtige Begriffe erläutert:

Globalisierung Eine breite Palette an technischen und geschäftlichen Prozessen, die erforderlich sind, um Produkte und Unternehmensaktivitäten für den globalen Markt vorzubereiten und in verschiedenen Ländern einzuführen. Die Globalisierung umfasst technische Prozesse wie Internationalisierung, Lokalisierung und Kulturalisierung sowie Geschäftsprozesse wie Produktmanagement, Finanzplanung, Marketing und Rechtsfragen. Globalisierung wird manchmal mit *G11n* abgekürzt (Buchstabe G, 11 weitere Buchstaben und dann der Buchstabe n für den englischen Begriff „Globalization“). *„Globalisierung ist Arbeitsthema von Unternehmen.“*

Internationalisierung Ein technischer Prozess zur allgemeinen Gestaltung eines Produkts, damit dieses mehrere Sprachen, Skripts und kulturelle Konventionen (wie Währungen, Sortierregeln, Zahlen- und Datumsformate) verarbeiten kann, ohne dass Design und Kompilierung neu durchgeführt werden müssen. Dieser Prozess lässt sich in zwei Arbeitsaufgaben unterteilen: Konzeptumsetzung und Lokalisierung. Die Internationalisierung ist auch als „globale Bereitschaft“ oder unter dem englischen Begriff *World-Readiness* bekannt und wird manchmal mit *I18n* abgekürzt. *„Internationalisierung ist Arbeitsthema von Ingenieuren.“*

Lokalisierung Die Anpassung von Produkten oder Dienstleistungen an eine bestimmte Sprache und Kultur, damit sie ein passendes Erscheinungsbild für den Zielmarkt aufweisen. Der Begriff „Lokalisierung“ wird manchmal mit *L10n* abgekürzt. *„Lokalisierung ist Arbeitsthema von Übersetzern.“*

Kulturalisierung Ein technischer Prozess, mit dem bestimmte Merkmale an die spezifischen Anforderungen eines bestimmten Kulturkreises angepasst werden. Beispiele hierfür sind die japanischen Veröffentlichungsfunktionen in Adobe InDesign sowie die Hanko-Unterstützung in Adobe Acrobat.

Im Folgenden werden einige weitere wichtige Begriffe in Bezug auf die Internationalisierung definiert:

Zeichensatz Die Zeichen, die für die Schrift in einer oder mehreren Sprachen verwendet werden. Ein Zeichensatz umfasst sprachspezifische Zeichen, Sonderzeichen (wie Interpunktionszeichen und mathematische Symbole), Ziffern und Computersteuerungszeichen.

Sortieren Das Sortieren von Text in der richtigen Reihenfolge für ein bestimmtes Gebietsschema.

Ländereinstellung Zum Gebietsschema zählen die sprachlichen und kulturellen Konventionen, die in einer geografischen, politischen oder kulturellen Region üblich sind (häufig handelt es sich dabei um ein bestimmtes Land). Ein eindeutiger Bezeichner für das Gebietsschema (Gebietsschema-ID) repräsentiert diesen Wert. Diese Gebietsschema-ID wird zum Nachschlagen verschiedener Daten zum Gebietsschema verwendet, die Unterstützung für ein bestimmtes Gebietsschema bieten. Zu dieser Unterstützung zählen Maßeinheiten, die Analyse und Formatierung von Zahlen und Datumsangaben und so weiter.

Ressourcenpaket Ein gespeicherter Satz an Elementen, die konkret für das Gebietsschema erstellt werden, in dem eine Anwendung eingesetzt wird. Ein Ressourcenpaket enthält normalerweise alle Textelemente der Benutzeroberfläche der Anwendung. Innerhalb des Pakets werden diese Elemente in die entsprechende Sprache des angegebenen Gebietsschemas übersetzt. Außerdem kann es weitere Einstellungen enthalten, die das Layout oder Verhalten der Benutzeroberfläche an ein bestimmtes Gebietsschema anpassen. Ein Ressourcenpaket kann andere Medientypen oder Verweise auf andere Medientypen enthalten, die sich konkret auf ein Gebietsschema beziehen.

Übersicht über das flash.globalization-Paket

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Das flash.globalization-Paket nutzt die Funktionsmerkmale für die kulturelle Unterstützung, die vom zugrunde liegenden Betriebssystem bereitgestellt werden. Es vereinfacht die Entwicklung von Anwendungen, die den kulturellen Konventionen individueller Benutzer entsprechen.

Das Paket enthält die folgenden Hauptklassen:

- Die Collator-Klasse regelt das Sortieren und Zuordnen von Strings.
- Die CurrencyFormatter-Klasse formatiert Zahlen in Strings mit Währungsbeträgen und analysiert Währungsbeträge und Symbole in den Eingabestrings.
- Die DateTimeFormatter-Klasse formatiert Datumswerte.
- Die LocaleID-Klasse ruft Informationen über ein bestimmtes Gebietsschema ab.
- Die NumberFormatter-Klasse formatiert und analysiert numerische Werte.
- Die StringTools-Klasse handhabt die an das Gebietsschema angepasste Umwandlung der Groß- und Kleinschreibung von Strings.

flash.globalization-Paket und Ressourcenlokalisierung

Das flash.globalization-Paket führt keine Lokalisierung der Ressourcen durch. Sie können jedoch die Gebietsschema-ID von flash.globalization als Schlüsselwert zum Abrufen lokalisierter Ressourcen mithilfe von anderen Techniken verwenden. Beispielsweise können Sie mit Flex erstellte Anwendungsressourcen mit den ResourceManager- und ResourceBundle-Klassen lokalisieren. Weitere Informationen finden Sie unter Localizing Flex Applications.

Adobe AIR 1.1 bietet auch einige Funktionsmerkmale für die Lokalisierung von AIR-Anwendungen, wie unter „Lokalisieren von AIR-Anwendungen“ auf Seite 1025 beschrieben.

Allgemeine Strategie zur Internationalisierung einer Anwendung

Im Folgenden wird in groben Zügen ein gängiges Verfahren zum Internationalisieren einer Anwendung mit dem flash.globalization-Paket beschrieben:

1. Bestimmen Sie das Gebietsschema.
2. Erstellen Sie eine Instanz einer Dienstklasse (Collator, CurrencyFormatter, DateTimeFormatter, NumberFormatter oder StringTools).
3. Ermitteln Sie Fehler und Ausweichsituationen mithilfe der lastOperationStatus-Eigenschaften.
4. Verwenden Sie spezifische Einstellungen für das Gebietsschema, um Informationen zu formatieren und anzuzeigen.

Im nächsten Schritt werden Strings und Benutzeroberflächen-Ressourcen, die sich konkret auf das Gebietsschema beziehen, geladen und angezeigt. Dieser Schritt kann beispielsweise folgende Aufgaben umfassen:

- Verwenden der automatischen Layoutfunktionen zum Anpassen der Größe der Benutzeroberfläche an die Stringlängen
- Auswahl der passenden Schriftarten und Unterstützung von Alternativschriftarten
- Verwenden der FTE-Textengine zur Unterstützung anderer Schreibsysteme
- Gewährleisten der richtigen Verwendung von Eingabemethoden-Editoren

Ermitteln von Fehlern und Ausweichsituationen

Die flash.globalization-Dienstklassen folgen alle einem ähnlichen Muster zum Identifizieren von Fehlern. Auch wenn das angeforderte Gebietsschema nicht verfügbar ist, gilt ein ähnliches Muster zum Ausweichen auf ein Gebietsschema, das vom Betriebssystem des Benutzers unterstützt wird.

Das folgende Beispiel zeigt, wie Sie beim Instanziieren von Dienstklassen Fehler und Ausweichsituationen erkennen können. Jede Dienstklasse hat eine lastOperationStatus-Eigenschaft, die angibt, ob der zuletzt durchgeführte Methodenaufruf die Fehler oder Warnungen ausgelöst hat.

```
var nf:NumberFormatter = new NumberFormatter("de-DE");
if(nf.lastOperationStatus != LastOperationStatus.NO_ERROR)
{
    if(nf.lastOperationStatus == LastOperationStatus.USING_FALLBACK_WARNING)
    {
        // perform fallback logic here, if needed
        trace("Warning - Fallback locale ID: " + nf.actualLocaleIDName);
    }
    else
    {
        // perform error handling logic here, if needed
        trace("Error: " + nf.lastOperationStatus);
    }
}
```

In diesem Beispiel wird eine Meldung aufgezeichnet, wenn ein Fehler auftritt oder wenn eine Gebietsschema-ID als Ausweichlösung verwendet wird. Ihre Anwendung kann bei Bedarf zusätzliche Logik zur Fehlerverarbeitung ausführen. Beispielsweise können Sie eine Meldung für den Benutzer einblenden oder die Anwendung zwingen, ein bestimmtes unterstütztes Gebietsschema zu verwenden.

Bestimmen des Gebietsschemas

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Ein Gebietsschema definiert eine bestimmte Kombination aus Sprache und kulturellen Gebräuchen für ein Land oder eine Region.

Eine Gebietsschema-ID kann auf sichere Weise als String verwaltet werden. Sie können jedoch die LocaleID-Klasse verwenden, um zusätzliche Informationen zu einem Gebietsschema abzurufen.

Ein LocaleID-Objekt wird folgendermaßen erstellt:

```
var locale:LocaleID = new LocaleID("es-MX");
```

Nachdem das LocaleID-Objekt erstellt wurde, können Sie Daten über die Gebietsschema-ID abrufen. Verwenden Sie die Methoden `getKeysAndValues()`, `getLanguage()`, `getRegion()`, `getScript()`, `getVariant()` und `isRightToLeft()` sowie die `name`-Eigenschaft.

Die über diese Methoden und Eigenschaften abgerufenen Werte können zusätzliche Informationen zum Gebietsschema enthalten, die nicht direkt aus der Gebietsschema-ID extrahiert werden können.

Wenn eine Anwendung einen Dienst erstellt, der das Gebietsschema erkennt (wie beispielsweise zur Datumsformatierung), muss das vorgesehene Gebietsschema angegeben werden. Die Liste der unterstützten Gebietsschemas variiert von Betriebssystem zu Betriebssystem. Deshalb ist es möglich, dass das angeforderte Gebietsschema nicht zur Verfügung steht.

Flash Player versucht zunächst, den Sprachcode des angeforderten Gebietsschemas zuzuordnen. Dann versucht Flash Player, das Gebietsschema genauer zu definieren, indem ein passendes Schreibsystem (Skript) und eine zugehörige Region gesucht werden. Zum Beispiel:

```
var loc:LocaleID = new LocaleID("es");
trace(loc.getLanguage()); // es
trace(loc.getScript()); // Latn
trace(loc.getRegion()); // ES
```

In diesem Beispiel hat der `LocaleID()`-Konstruktor Daten über das Gebietsschema abgerufen, das dem Sprachcode „es“ für diesen Benutzer am besten entspricht.

Einstellen der Gebietsschema-ID

Das aktuelle Gebietsschema für eine Anwendung lässt sich mit verschiedenen Verfahren einstellen, wie zum Beispiel:

- Programmieren Sie eine einzelne, hartkodierte Gebietsschema-ID in der Anwendung. Dies ist ein übliches Verfahren, das jedoch die Internationalisierung der Anwendung nicht unterstützt.
- Verwenden Sie die Voreinstellungen für die Gebietsschema-ID, die im Betriebssystem oder Browser des Benutzers oder in anderen Benutzervoreinstellungen festgelegt sind. Durch dieses Verfahren werden meist die besten Gebietsschema-Einstellungen für den Benutzer erzielt, das Verfahren ist jedoch nicht immer ganz genau. Es besteht das Risiko, dass die Einstellungen des Betriebssystems den tatsächlichen Voreinstellungen des Benutzers nicht entsprechen. Dies ist beispielsweise der Fall, wenn der Benutzer seinen Computer mit anderen Personen gemeinsam nutzt und die bevorzugten Gebietsschemas des Betriebssystems nicht ändern kann.
- Nachdem Sie die Gebietsschema-ID auf Grundlage der Benutzervoreinstellungen festgelegt haben, geben Sie dem Benutzer die Möglichkeit, in einer Liste der unterstützten Gebietsschemas eine Auswahl zu treffen. Diese Strategie ist normalerweise die beste Option, wenn Ihre Anwendung mehr als ein Gebietsschema unterstützt.

Diese dritte Option kann folgendermaßen implementiert werden:

1 Rufen Sie eine Liste der bevorzugten Gebietsschemas oder Sprachen des Benutzers aus einem Benutzerprofil, aus den Einstellungen des Browsers oder Betriebssystems oder von einem Cookie ab. (Ihre Anwendung muss diese Logik selbst implementieren. Die flash.globalization-Bibliothek bietet keine Unterstützung für das direkte Lesen dieser Voreinstellungen.)

2 Stellen Sie fest, welche dieser Gebietsschemas von Ihrer Anwendung unterstützt werden, und wählen Sie das beste Gebietsschema als Standardeinstellung aus. Verwenden Sie die LocaleID.determinePreferredLocales()-Methode, um die besten Gebietsschemas für einen Benutzer anhand seines bevorzugten Gebietsschemas und der vom Betriebssystem unterstützten Gebietsschemas zu ermitteln.

3 Richten Sie für den Benutzer eine Möglichkeit ein, das Standardgebietsschema zu ändern, falls es nicht zufriedenstellend ist.

Einschränkungen anderer Klassen für Gebietsschemas und Sprachen

Mit der `fl.lang.Locale`-Klasse können Sie Textstrings auf Grundlage eines Gebietsschemas ersetzen und dazu Ressourcenpakete mit Stringwerten verwenden. Diese Klasse bietet jedoch keine Unterstützung für andere Internationalisierungsmerkmale, wie beispielsweise Formatierung von Zahlen, Währungen und Datumsangaben sowie Sortierung und Zuordnung. Außerdem steht diese Klasse nur in Flash Professional zur Verfügung.

Die aktuelle Sprachcode-Einstellung für das Betriebssystem kann auch über die `flash.system.Capabilities.language`-Eigenschaft abgerufen werden. Diese Eigenschaft ruft jedoch nur den zweistelligen ISO 639-1-Sprachcode auf, nicht die vollständige Gebietsschema-ID. Außerdem unterstützt sie nur bestimmte Gebietsschemas.

In AIR 1.5 können Sie die `flash.system.Capabilities.languages`-Eigenschaft verwenden. Diese Eigenschaft stellt ein Array der vom Benutzer bevorzugten Sprachen für die Benutzeroberfläche bereit. Deshalb unterliegt sie nicht den Einschränkungen von `Capabilities.language`.

Formatieren von Zahlen

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Das Format für die Anzeige numerischer Werte variiert stark je nach Region. Im Folgenden wird als Beispiel gezeigt, wie die Zahl 123456.78 in verschiedenen Gebietsschemas formatiert wird:

Ländereinstellung	Zahlenformat
en-US (Englisch, USA)	-123,456.78
de-DE (Deutsch, Deutschland)	-123.456,78
fr-FR (Französisch, Frankreich)	-123 456,78
de-CH (Deutsch, Schweiz)	-123'456.78
en-IN (Englisch, Indien)	-1,23,456.78
Zahlreiche arabische Gebietsschemas	123,456.78-

Zahlreiche Faktoren wirken sich auf die Zahlenformate aus, wie zum Beispiel:

- Trennzeichen. Das Dezimaltrennzeichen befindet sich zwischen der Ganzzahl und dem Bruchteil einer Zahl. Es kann sich dabei um ein Komma, einen Punkt oder um ein anderes Zeichen handeln. Das Gruppentrennzeichen (Tausendertrennzeichen) kann ein Komma, ein Punkt, ein geschütztes Leerzeichen oder ein anderes Zeichen sein.
- Gruppenmuster. Zwischen jedem Gruppentrennzeichen links vom Dezimaltrennzeichen können zwei, drei oder eine andere Anzahl Ziffern stehen.
- Zeichen für negative Werte. Negative Werte können mit einem Minuszeichen links oder rechts von der Zahl gekennzeichnet sein. Im Finanzwesen stehen negative Werte häufig in Klammern. Die negative Zahl 19 kann beispielsweise folgendermaßen dargestellt werden: -19, 19- oder (19).
- Führende und nachgestellte Nullen. In einigen regionalen Konventionen werden Zahlen mit führenden oder nachgestellten Nullen angezeigt. So sind für den Wert 0.17 unter anderem folgende Darstellungen möglich: .17, 0.17 oder 0.170.
- Ziffernzeichensätze. Viele Sprachen, darunter Hindi, Arabisch und Japanisch, verwenden verschiedene Ziffernzeichensätze. Das flash.globalization-Paket unterstützt Ziffernzeichensätze, die den Ziffern 0-9 zugeordnet werden können.

Die NumberFormatter-Klasse berücksichtigt all diese Faktoren bei der Formatierung von numerischen Werten.

Verwenden der NumberFormatter-Klasse

Die NumberFormatter-Klasse formatiert numerische Werte (mit den int-, uint- und Number-Datentypen) gemäß den Konventionen eines bestimmten Gebietsschemas.

Das folgende Beispiel zeigt das einfachste Verfahren zur Formatierung einer Zahl unter Verwendung der standardmäßigen Formatierungseigenschaften, die vom Betriebssystem des Benutzers bereitgestellt werden:

```
var nf:NumberFormatter = new NumberFormatter(LocaleID.DEFAULT);
trace(nf.formatNumber(-123456.789))
```

Das Ergebnis richtet sich nach den Gebietsschema-Einstellungen und anderen Voreinstellungen des Benutzers. Wenn der Benutzer zum Beispiel das Gebietsschema fr-FR gewählt hat, wird der Wert folgendermaßen formatiert:

-123.456,789

Wenn Sie eine Zahl unabhängig von der Benutzereinstellung für ein bestimmtes Gebietsschema formatieren möchten, geben Sie den Namen des Gebietsschemas explizit an. Zum Beispiel:

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
trace(nf.formatNumber(-123456.789));
```

Das Ergebnis lautet:

-123'456.789

Die formatNumber()-Methode akzeptiert eine Zahl mit dem Number-Datentyp als Parameter. Außerdem enthält die NumberFormatter-Klasse die formatInt()-Methode, die den int-Datentyp als Eingabe akzeptiert, und die formatUint()-Methode, die den uint-Datentyp akzeptiert.

Sie können die Formatierungslogik explizit steuern, indem Sie die Eigenschaften der NumberFormatter-Klasse festlegen, wie im folgenden Beispiel gezeigt:

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
nf.negativeNumberFormat = 0;
nf.fractionalDigits = 5;
nf.trailingZeros = true;
nf.decimalSeparator = ",";
nf.useGrouping = false;
trace(nf.formatNumber(-123456.789)); //(123456.78900)
```

In diesem Beispiel wird zunächst ein NumberFormatter-Objekt erstellt. Dann werden die folgenden Schritte ausgeführt:

- Das Format für negative Zahlen wird auf Klammern eingestellt (wie für Finanzanwendungen).
- Die Anzahl der Stellen nach dem Dezimaltrennzeichen wird auf 5 eingestellt.
- Über nachgestellte Nullen wird festgelegt, dass fünf Dezimalstellen angezeigt werden.
- Als Dezimaltrennzeichen wird das Komma festgelegt.
- Die Formatierungslogik wird angewiesen, keine Gruppentrennzeichen zu verwenden.

Hinweis: *Wenn einige dieser Eigenschaften sich ändern, entspricht das resultierende Zahlenformat nicht mehr dem bevorzugten Format für das angegebene Gebietsschema. Einige dieser Eigenschaften sollten nur verwendet werden, wenn die Erkennung des Gebietsschemas nicht von Bedeutung ist, wenn ein bestimmter Aspekt des Formats, wie die Anzahl der nachgestellten Nullen, genau gesteuert werden muss, oder wenn der Benutzer die Änderung direkt anfordert, beispielsweise über die Windows-Systemsteuerung.*

Analysieren von Strings mit numerischen Werten

Die NumberFormatter-Klasse kann auch numerische Werte aus Strings extrahieren, die den Formatierungsanforderungen eines bestimmten Gebietsschemas entsprechen. Die NumberFormatter.parseNumber()-Methode extrahiert einen einzelnen numerischen Wert aus einem String. Zum Beispiel:

```
var nf:NumberFormatter = new NumberFormatter( "en-US" );
var inputNumberString:String =  "-1,234,567.890"
var parsedNumber:Number = nf.parseNumber(inputNumberString);
trace("Value:" + parsedNumber); // -1234567.89
trace("Status:" + nf.lastOperationStatus); // noError
```

Die parseNumber()-Methode verarbeitet Strings, die nur Ziffern und Zeichen für die Zahlenformatierung enthalten, wie Zeichen für negative Zahlen und Trennzeichen. Wenn der String andere Zeichen enthält, wird ein Fehlercode festgelegt:

```
var nf:NumberFormatter = new NumberFormatter( "en-US" );
var inputNumberString:String =  "The value is 1,234,567.890"
var parsedNumber:Number = nf.parseNumber(inputNumberString);
trace("Value:" + parsedNumber); // NaN
trace("Status:" + nf.lastOperationStatus); // parseError
```

Zum Extrahieren von Zahlen aus Strings, die auch alphabetische Zeichen enthalten, verwenden Sie die NumberFormatter.parse()-Methode:

```
var nf:NumberFormatter = new NumberFormatter( "en-US" );
var inputNumberString:String = "The value is 123,456,7.890";
var parseResult:NumberParseResult = nf.parse(inputNumberString);
trace("Value:" + parseResult.value); // 1234567.89
trace("startIndex: " + parseResult.startIndex); // 14
trace("Status:" + nf.lastOperationStatus); // noError
```

Die parse()-Methode gibt ein NumberParseResult-Objekt zurück, das den analysierten numerischen Wert in seiner Werteigenschaft enthält. Die startIndex-Eigenschaft gibt den Index des ersten gefundenen numerischen Zeichens an. Mithilfe der startIndex- und endIndex-Eigenschaften können Sie die Teile des Strings extrahieren, die vor und nach den Ziffern stehen.

Formatieren von Währungswerten

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Die Anzeigeformate von Währungen unterscheiden sich ebenso stark wie die Zahlenformate. Im Folgenden wird als Beispiel gezeigt, wie der US-Dollar-Wert $123456.78 in verschiedenen Gebietsschemas formatiert wird:

Ländereinstellung	Zahlenformat
en-US (Englisch, USA)	$123,456.78
de-DE (Deutsch, Deutschland)	123.456,78 $
en-IN (Englisch, Indien)	$ 1,23,456.78

Bei der Währungsformatierung gelten dieselben Faktoren wie bei der Zahlenformatierung, zusätzlich jedoch auch die folgenden Aspekte:

- ISO-Code der Währung. Der dreistellige ISO 4217-Währungscode für das verwendete Gebietsschema, wie beispielsweise USD oder EUR.
- Währungssymbol. Das Symbol oder der String der Währung im verwendeten Gebietsschema, wie $ oder €.
- Format für negative Währungsbeträge. Definiert die Position des Währungssymbols und des Symbols für negative Beträge oder Klammern in Bezug auf den numerischen Teil des Währungsbetrags.
- Format für positive Währungsbeträge. Definiert die Position des Währungssymbols in Bezug auf den numerischen Teil des Währungsbetrags.

Verwenden der CurrencyFormatter-Klasse

Die CurrencyFormatter-Klasse formatiert numerische Werte in Strings, die Währungsstrings und formatierte Zahlen enthalten, und zwar gemäß den Konventionen eines bestimmten Gebietsschemas.

Wenn Sie ein neues CurrencyFormatter-Objekt instanziieren, stellt es die Währung auf die Standardwährung des angegebenen Gebietsschemas ein.

Das folgende Beispiel zeigt, dass ein mit dem deutschen Gebietsschema erstelltes CurrencyFormatter-Objekt automatisch Währungsbeträge in Euro verwendet:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "de-DE" );
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

In den meisten Fällen sollten Sie sich nicht auf die Standardwährung für ein Gebietsschema verlassen. Wenn das Standardgebietsschema des Benutzers nicht unterstützt wird, weist die CurrencyFormatter-Klasse ein anderes Gebietsschema als Ausweichlösung zu. Dieses alternative Gebietsschema kann eine andere Standardwährung aufweisen. Außerdem soll in der Regel sichergestellt werden, dass die Währungsformate korrekt für den Benutzer dargestellt werden, auch wenn die Beträge nicht in der Währung des Benutzers angezeigt werden. So sollen die Preise eines deutschen Unternehmens für einen kanadischen Benutzer zwar in Euro, aber im kanadischen Format angezeigt werden.

Die CurrencyFormatter.setCurrency()-Methode gibt genau an, welcher Währungsstring und welches Währungssymbol verwendet werden sollen.

Der Code im folgenden Beispiel zeigt Euro-Beträge für Benutzer im französischsprachigen Teil Kanadas an.

```
var cf:CurrencyFormatter = new CurrencyFormatter( "fr-CA" );
cf.setCurrency("EUR", "€");
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

Mithilfe der setCurrency()-Methode können auch eindeutige Währungssymbole festgelegt werden, um Verwirrung zu vermeiden: Zum Beispiel:

```
cf.setCurrency("USD","US$");
```

Standardmäßig zeigt die format()-Methode einen dreistelligen ISO 4217-Währungscode anstelle des Währungssymbols an. ISO 4217-Codes sind eindeutig und müssen nicht lokalisiert werden. Viele Benutzer ziehen es jedoch vor, anstelle der ISO-Codes die Währungssymbole zu sehen.

Die CurrencyFormatter-Klasse hilft Ihnen bei der Entscheidung, welches Symbol ein formatierter Währungsstring enthalten soll, wie das Dollar- oder Euro-Zeichen oder einen dreistelligen ISO-Währungsstring, wie USD oder EUR. Beispielsweise kann ein Wert in kanadischen Dollar für Benutzer in Kanada im Format $200 angezeigt werden. Für Benutzer in den USA wird dagegen CAD 200 angezeigt. Mithilfe der formattingWithCurrencySymbolIsSafe()-Methode können Sie feststellen, ob das Währungssymbol eines Betrags bei der Gebietsschema-Einstellung des Benutzers unklar oder falsch wäre.

Im folgenden Beispiel wird ein Euro-Wert für das Gebietsschema „en-US“ formatiert. Je nach Gebietsschema des Benutzers enthält der Ausgabestring entweder den ISO-Währungscode oder das Währungssymbol.

```
var cf:CurrencyFormatter = new CurrencyFormatter( "en-CA");

if (cf.formattingWithCurrencySymbolIsSafe("USD"))
{
    trace(cf.format(1234567.89, true)); // $1,234,567.89
}
else
{
    cf.setCurrency("USD", "$");
    trace(cf.format(1234567.89)); // USD1,234,567.89
}
```

Analysieren von Strings mit Währungswerten

Die CurrencyFormatter-Klasse kann auch einen Währungsbetrag und einen Währungsstring aus einem Eingabestring extrahieren, der den Formatierungsanforderungen eines bestimmten Gebietsschemas entspricht. Die CurrencyFormatter.parse()-Methode speichert den analysierten Betrag und den Währungsstring in einem CurrencyParseResult-Objekt, wie im Folgenden gezeigt:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "en-US" );
var inputCurrencyString:String = "(GBP 123,56,7.890)";
var parseResult:CurrencyParseResult = cf.parse(inputCurrencyString);
trace("parsed amount: " + parseResult.value); // -1234567.89
trace("currencyString: " + parseResult.currencyString ); // GBP
```

Der Währungsstring des Eingabestrings kann ein Währungssymbol, einen ISO-Währungscode und zusätzliche Textzeichen enthalten. Die Positionen des Währungsstrings, des Zeichens für negative Zahlen und des numerischen Wertes entsprechen den Formaten, die von den negativeCurrencyFormat- und positiveCurrencyFormat-Eigenschaften angegeben sind. Zum Beispiel:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "en-US" );
var inputCurrencyString:String = "Total $-123,56,7.890";
var parseResult:CurrencyParseResult = cf.parse(inputCurrencyString);
trace("status: " + cf.lastOperationStatus ); // parseError
trace("parsed amount: " + parseResult.value); // NaN
trace("currencyString: " + parseResult.currencyString ); //
cf.negativeCurrencyFormat = 2;
parseResult = cf.parse(inputCurrencyString);
trace("status: " + cf.lastOperationStatus ); // noError
trace("parsed amount: " + parseResult.value); // -123567.89
trace("currencyString: " + parseResult.currencyString ); // Total $
```

In diesem Beispiel besteht der Eingabestring aus einem Währungsstring gefolgt von einem Minuszeichen und einer Zahl. Der negativeCurrencyFormat-Standardwert für das Gebietsschema „en-US“ gibt jedoch vor, dass das Negativzeichen zuerst steht. Deshalb gibt die parse()-Methode einen Fehler aus und der analysierte Wert lautet NaN.

Nachdem negativeCurrencyFormat auf 2 eingestellt wurde, wodurch der Währungsstring zuerst steht, kann die parse()-Methode erfolgreich ausgeführt werden.

Formatieren von Datum und Uhrzeit

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Das Format zur Anzeige von Datum und Uhrzeit unterscheidet sich ebenfalls stark von Region zu Region. Im Folgenden wird beispielsweise gezeigt, wie der 2. Januar 1962, 13.01 Uhr, in Kurzform für bestimmte Gebietsschemas angezeigt wird:

Ländereinstellung	Format von Datum und Uhrzeit
en-US (Englisch, USA)	1/2/62 1:01pm
fr-FR (Französisch, Frankreich)	2/1/62 13:01
ja-JP (Japanisch, Japan)	1962/2/1 13:01

Verwenden der DateTimeFormatter-Klasse

Die DateTimeFormatter-Klasse formatiert Werte mit dem Date-Datentyp in Strings mit Datum und Uhrzeit, und zwar gemäß den Konventionen eines bestimmten Gebietsschemas.

Die Formatierung folgt einem Musterstring, der Buchstabenfolgen enthält, die durch ein Datum oder eine Uhrzeit ersetzt werden. Im Muster „jjjj/MM“ werden beispielsweise die Buchstaben „jjjj“ durch die vierstellige Jahreszahl ersetzt, danach folgt ein Schrägstrich (/) und eine zweistellige Monatsangabe.

Der Musterstring kann explizit über die setDateTimePattern()-Methode festgelegt werden. Es empfiehlt sich jedoch, das Muster automatisch über die Gebietsschema-Einstellungen des Benutzers und die Voreinstellungen des Betriebssystems festlegen zu lassen. So wird gewährleistet, dass das Ergebnis in kultureller Hinsicht geeignet ist.

Mithilfe von DateTimeFormatter können Datums- und Zeitangaben in drei Standardformaten angezeigt werden (LONG, MEDIUM und SHORT). Auch ein benutzerdefiniertes Muster (CUSTOM) kann verwendet werden. Es ist auch möglich, ein Format für das Datum und ein anderes Format für die Uhrzeit zu verwenden. Die tatsächlich für die einzelnen Formate verwendeten Muster sind je nach Betriebssystem etwas unterschiedlich.

Sie können die Formate angeben, wenn Sie ein DateTimeFormatter-Objekt erstellen. Wenn keine Formatparameter angegeben werden, wird der Standardwert DateTimeStyle.LONG verwendet. Sie können die Formate später über die setDateTimeStyles()-Methode ändern, wie im folgenden Beispiel gezeigt:

```
var date:Date = new Date(2009, 2, 27, 13, 1);
var dtf:DateTimeFormatter = new DateTimeFormatter("en-US",
    DateTimeStyle.LONG, DateTimeStyle.LONG);

var longDate:String = dtf.format(date);
trace(longDate); // March 27, 2009 1:01:00 PM

dtf.setDateTimeStyles(DateTimeStyle.SHORT, DateTimeStyle.SHORT);
var shortDate:String = dtf.format(date);
trace(shortDate); // 3/27/09 1:01 PM
```

Lokalisieren der Namen von Monaten und Tagen

In vielen Anwendungen werden Listen mit den Namen der Monate und Wochentage für die Anzeige in Kalendern und Pulldownlisten verwendet.

Über die DateTimeFormatter.getMonthNames()-Methode können Sie eine lokalisierte Liste der Monatsnamen abrufen. Je nach Betriebssystem stehen möglicherweise vollständige und abgekürzte Namen zur Verfügung. Für Monatsnamen in voller Länge übergeben Sie den Wert DateTimeNameStyle.FULL. Für eine Kurzversion übergeben Sie die Werte DateTimeNameStyle.LONG_ABBREVIATION oder DateTimeNameStyle.SHORT_ABBREVIATION.

In einigen Sprachen ändert sich der Name eines Monats in seine Genitivform, wenn er im Datumsformat neben dem Tag platziert wird. Wenn Sie vorhaben, nur den Namen des Monats zu verwenden, übergeben Sie deshalb den Wert DateTimeNameContext.STANDALONE an die getMonthNames()-Methode. Um die Monatsnamen in formatierten Datumsangaben zu verwenden, übergeben Sie jedoch den Wert DateTimeNameContext.FORMAT.

```
var dtf:DateTimeFormatter = new DateTimeFormatter("fr-FR");
var months:Vector.<String> = dtf.getMonthNames(DateTimeNameStyle.FULL,
        DateTimeNameContext.STANDALONE);
trace(months[0]); // janvier
months = dtf.getMonthNames(DateTimeNameStyle.SHORT_ABBREVIATION,
         DateTimeNameContext.STANDALONE);
trace(months[0]); // janv.
```

Die DateTimeFormatter.getWeekdayNames()-Methode liefert eine lokalisierte Liste der Namen der Wochentage. Die getWeekdayNames()-Methode akzeptiert dieselben nameStyle- und context-Parameter wie die getMonthNames()-Methode.

```
var dtf:DateTimeFormatter = new DateTimeFormatter("fr-FR");
var weekdays:Vector.<String> = dtf.getWeekdayNames(DateTimeNameStyle.FULL,
           DateTimeNameContext.STANDALONE);
trace(weekdays[0]); // dimanche
weekdays = dtf.getWeekdayNames(DateTimeNameStyle.LONG_ABBREVIATION,
           DateTimeNameContext.STANDALONE);
trace(weekdays[0]); // dim.
```

Außerdem gibt die getFirstWeekday()-Methode den Indexwert des Tages zurück, mit dem die Woche im ausgewählten Gebietsschema beginnt.

Sortieren und Vergleichen von Strings

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Beim Sortieren werden Elemente in ihre richtige Reihenfolge gebracht. Bei Sortierregeln bestehen zwischen verschiedenen Gebietsschemas große Unterschiede. Die Regeln unterscheiden sich auch für verschiedene Aufgaben, beispielsweise für das Sortieren einer Liste und für das Zuordnen von ähnlichen Elementen, wie bei einem Algorithmus für die Textsuche.

Beim Sortieren sind oft auch kleine Unterschiede von Bedeutung, wie Groß- und Kleinbuchstaben oder diakritische Zeichen wie Akzente oder Umlaute. So wird der Umlaut ö im Deutschen wie der Buchstabe o behandelt. Im Schwedischen steht er dagegen in der Sortierreihenfolge nach dem Buchstaben z. In Französisch und einigen anderen Sprachen bestimmt der letzte Akzentunterschied in einem Wort die Sortierreihenfolge.

Beim Suchen empfiehlt es sich oft, Unterschiede bei der Groß- und Kleinschreibung oder diakritische Zeichen zu ignorieren, um die Wahrscheinlichkeit zu erhöhen, dass Übereinstimmungen gefunden werden. Wenn Sie beispielsweise in einem französischen Dokument nach den Zeichen „cote“ suchen, werden möglicherweise zusätzlich zu „cote“ auch Übereinstimmungen mit „côte“ und „coté“ zurückgegeben.

Verwenden der Collator-Klasse

Die Hauptmethoden der Collator-Klasse sind die compare()-Methode, die hauptsächlich zum Sortieren verwendet wird, und die equals()-Methode zum Zuordnen von Werten.

Das folgende Beispiel zeigt das unterschiedliche Verhalten der compare()- und equals()-Methoden.

```
var words:Array = new  Array("coté", "côte");

var sorter:Collator = new Collator("fr-FR", CollatorMode.SORTING);
words.sort(sorter.compare);
trace(words); // côte,coté

var matcher:Collator = new Collator("fr-FR", CollatorMode.MATCHING);
if (matcher.equals(words[0], words[1]))
{
    trace(words[0] + " = " + words[1]); // côte = coté
}
```

Das Beispiel erstellt zunächst ein Collator-Objekt im SORTING-Modus für das Gebietsschema „Französisch-Frankreich“. Dann werden zwei Wörter sortiert, die sich nur hinsichtlich der diakritischen Zeichen unterscheiden. Dies zeigt, dass beim SORTING-Vergleich zwischen Buchstaben mit und ohne Akzentzeichen unterschieden wird.

Der Sortiervorgang erfolgt, indem ein Verweis auf die sort()-Methode des Collator-Objekts als Parameter an die Array.sort()-Methode übergeben wird. Dies ist eine der effizientesten Techniken, um die Sortierreihenfolge mit einem Collator-Objekt zu steuern.

Dann erstellt das Beispiel ein Collator-Objekt im MATCHING-Modus. Wenn das Collator-Objekt die beiden Wörter vergleicht, werden sie als gleich interpretiert. Dies zeigt, dass Zeichen mit Akzent beim MATCHING-Vergleich genauso behandelt werden wie Zeichen ohne Akzent.

Anpassen des Verhaltens der Collator-Klasse

Standardmäßig verwendet die Collator-Klasse Stringvergleichsregeln des Betriebssystems, die sich nach dem Gebietsschema und den Systemvoreinstellungen des Benutzers richten. Sie können das Verhalten der compare()- und equals()-Methoden anpassen, indem Sie explizit verschiedene Eigenschaften festlegen. In der folgenden Tabelle werden die Eigenschaften und ihre Auswirkungen auf Vergleiche aufgelistet:

Collator-Eigenschaft	Effekt
numericComparison	Steuert, ob Ziffern als Zahlen oder als Text behandelt werden.
ignoreCase	Steuert, ob Unterschiede zwischen Groß- und Kleinschreibung ignoriert werden.
ignoreCharacterWidth	Steuert, ob die Formen in halber oder voller Breite einiger chinesischer und japanischer Zeichen als gleich interpretiert werden.
ignoreDiacritics	Steuert, ob Strings, die sich nur hinsichtlich Akzentzeichen oder anderer diakritischer Zeichen unterscheiden, als gleich interpretiert werden.
ignoreKanaType	Steuert, ob Strings, die sich nur hinsichtlich des Typs des verwendeten Kana-Zeichens unterscheiden, als gleich interpretiert werden.
ignoreSymbols	Steuert, ob Symbolzeichen wie beispielsweise Leerzeichen, Währungssymbole und mathematische Symbole ignoriert werden.

Das folgende Codebeispiel zeigt, dass die Sortierreihenfolge in einer Liste mit französischen Wörtern sich ändert, wenn die ignoreDiacritics-Eigenschaft auf „true“ gesetzt wird:

```
var words:Array = new  Array("COTE", "coté", "côte", "Coté","cote");
var sorter:Collator = new Collator("fr-CA", CollatorMode.SORTING);
words.sort(sorter.compare);
trace(words); // cote,COTE,côte,coté,Coté

sorter.ignoreDiacritics = true;
words.sort(sorter.compare);
trace(words); // côte,coté,cote,Coté,COTE
```

Umwandlung der Groß- und Kleinschreibung

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

In verschiedenen Sprachen gelten auch unterschiedliche Regeln für die Umwandlung von Großbuchstaben in Kleinbuchstaben.

So gehört in den meisten Sprachen mit dem lateinischen Alphabet zum Großbuchstabe „I“ der Kleinbuchstabe „i“. In einigen Sprachen, wie zum Beispiel Türkisch und Aserbaidschanisch, gibt es aber zusätzlich auch den Kleinbuchstaben ohne Punkt „ı“. In diesen Sprachen wird der Kleinbuchstabe „ı“ (ohne Punkt) in den Großbuchstaben „I“ umgewandelt. Der Kleinbuchstabe „i“ (mit Punkt) wird dagegen in den Großbuchstaben „İ“ (ebenfalls mit Punkt) umgewandelt.

Die StringTools-Klasse bietet Methoden, die sprachspezifische Regeln für diese Umwandlungen verwenden.

Verwenden der StringTools-Klasse

Die StringTools-Klasse bietet zwei Methoden zum Umwandeln der Groß- und Kleinschreibung: toLowerCase() und toUpperCase(). Sie erstellen ein StringTools-Objekt, indem Sie den Konstruktor mit einer Gebietsschema-ID aufrufen. Die StringTools-Klasse ruft die Regeln zur Umwandlung von Groß- und Kleinschreibung für das jeweilige Gebietsschema (oder ein alternatives Gebietsschema als Ausweichlösung) vom Betriebssystem ab. Der Algorithmus zur Umwandlung von Groß- und Kleinschreibung kann nicht weiter angepasst werden.

Im folgenden Codebeispiel werden die toUpperCase()- und toLowerCase()-Methoden verwendet, um den Buchstaben „ß“ in einem Satz umzuwandeln.

```
var phrase:String = "Schloß Neuschwanstein";
var converter:StringTools = new StringTools("de-DE");

var upperPhrase:String = converter.toUpperCase(phrase);
trace(upperPhrase); // SCHLOSS NEUSCHWANSTEIN

var lowerPhrase:String = converter.toLowerCase(upperPhrase);
trace(lowerPhrase);// schloss neuschwanstein
```

Die toUpperCase()-Methode wandelt den Kleinbuchstaben „ß“ in zwei groß geschriebene „SS“ um. Diese Transformation erfolgt jedoch nur in einer Richtung. Wenn die Buchstaben „SS“ wieder in Kleinbuchstaben umgewandelt werden, ist das Ergebnis „ss“, nicht „ß“.

Beispiel: Internationalisierung einer Börsenticker-Anwendung

Flash Player 10.1 und höher, Adobe AIR 2.0 und höher

Mit der Anwendung „Global Stock Ticker“ werden fiktive Börsendaten aus drei verschiedenen Börsen (USA, Japan und Europa) abgerufen und angezeigt. Die Anwendung formatiert die Daten gemäß den Konventionen verschiedener Gebietsschemas.

Dieses Beispiel veranschaulicht die folgenden Funktionsmerkmale des flash.globalization-Pakets:

- Zahlenformatierung mit Erkennung des Gebietsschemas
- Währungsformatierung mit Erkennung des Gebietsschemas
- Einstellen von ISO-Währungscodes und Währungssymbolen
- Datumsformatierung mit Erkennung des Gebietsschemas
- Abrufen und Anzeigen der passenden Abkürzungen für Monatsnamen

Die Anwendungsdateien für dieses Beispiel finden Sie unter www.adobe.com/go/learn_programmingAS3samples_flash_de. Die Dateien für die Anwendung „Global Stock Ticker“ befinden sich im Ordner „Samples/GlobalStockTicker“. Die Anwendung umfasst die folgenden Dateien:

Datei	Beschreibung
GlobalStockTicker.mxml oder GlobalStockTicker.fla	Die Benutzeroberfläche der Anwendung im Flex-Format (MXML) oder Flash-Format (FLA).
styles.css	Stile für die Benutzeroberfläche der Anwendung (nur Flex).
com/example/programmingas3/stockticker/flex/FinGraph.mxml	Eine MXML-Komponente, die ein Diagramm mit simulierten Börsendaten anzeigt (nur Flex).
com/example/programmingas3/stockticker/flash/GlobalStockTicker.as	Document-Klasse mit der Benutzeroberflächenlogik für die Anwendung (nur Flash).
comp/example/programmingas3/stockticker/flash/RightAlignedColumn.as	Ein benutzerdefinierter Zellen-Renderer für die DataGrid-Komponente in Flash (nur Flash).
com/example/programmingas3/stockticker/FinancialGraph.as	Eine ActionScript-Klasse, die ein Diagramm mit simulierten Börsendaten zeichnet.
com/example/programmingas3/stockticker/Localizer.as	Eine ActionScript-Klasse, die das Gebietsschema und die Währung verwaltet und Zahlen, Währungsbeträge und Datumsangaben je nach Gebietsschema formatiert.
com/example/programmingas3/stockticker/StockDataModel.as	Eine ActionScript-Klasse, die alle Beispieldaten für die Beispielanwendung „Global Stock Ticker“ enthält.

Benutzeroberfläche und Beispieldaten

Die Benutzeroberfläche der Anwendung besteht aus den folgenden Hauptelementen:

- Kombinationsfeld zum Auswählen des Gebietsschemas
- Kombinationsfeld zum Auswählen einer Börse
- DataGrid-Objekt zum Anzeigen von Daten für sechs Unternehmen einer jeden Börse
- Diagramm mit simulierten historischen Daten zu den Wertpapieren des ausgewählten Unternehmens

Alle Beispieldaten über Gebietsschemas, Börsen und Wertpapiere der Anwendung werden in der StockDataModel-Klasse gespeichert. Eine echte Anwendung würde Daten von einem Server abrufen und dann in einer Klasse wie StockDataModel speichern. In diesem Beispiel sind alle Daten in der StockDataModel-Klasse hartkodiert.

Hinweis: *Die Daten, die im Finanzdiagramm angezeigt werden, stimmen nicht unbedingt mit den Daten im DataGrid-Objekt überein. Bei Auswahl eines anderen Unternehmens wird das Diagramm immer zufällig neu gezeichnet. Es dient nur zu Illustrationszwecken.*

Einstellen des Gebietsschemas

Nachdem einige Schritte zur Einrichtung durchgeführt wurden, ruft die Anwendung die Localizer.setLocale()-Methode auf, um Formatter-Objekte für das Standardgebietsschema zu erstellen. Die setLocale()-Methode wird auch immer aufgerufen, wenn der Benutzer einen neuen Wert aus dem Kombinationsfeld für das Gebietsschema auswählt.

```
public function setLocale(newLocale:String):void
{
    locale  = new LocaleID(newLocale);

    nf = new NumberFormatter(locale.name);
    traceError(nf.lastOperationStatus, "NumberFormatter", nf.actualLocaleIDName);

    cf = new CurrencyFormatter(locale.name);
    traceError(cf.lastOperationStatus, "CurrencyFormatter", cf.actualLocaleIDName);
    symbolIsSafe = cf.formattingWithCurrencySymbolIsSafe(currentCurrency);
    cf.setCurrency(currentCurrency, currentSymbol);
    cf.fractionalDigits = currentFraction;

    df = new DateTimeFormatter(locale.name, DateTimeStyle.LONG, DateTimeStyle.SHORT);
    traceError(df.lastOperationStatus, "DateTimeFormatter", df.actualLocaleIDName);
    monthNames = df.getMonthNames(DateTimeNameStyle.LONG_ABBREVIATION);
}

public function traceError(status:String, serviceName:String, localeID:String) :void
{
    if(status != LastOperationStatus.NO_ERROR)
    {
        if(status == LastOperationStatus.USING_FALLBACK_WARNING)
        {
            trace("Warning - Fallback locale ID used by "
                    + serviceName + ": " + localeID);
        }
        else if (status == LastOperationStatus.UNSUPPORTED_ERROR)
        {
            trace("Error in " + serviceName + ": " + status);
            //abort application
            throw(new Error("Fatal error", 0));
        }
        else
        {
            trace("Error in " + serviceName + ": " + status);
        }
    }
    else
    {
        trace(serviceName + " created for locale ID: " + localeID);
    }
}
```

Zunächst erstellt die setLocale()-Methode ein LocaleID-Objekt. Bei Verwendung dieses Objekts ist es später einfacher, bei Bedarf Einzelheiten zum tatsächlichen Gebietsschema abzurufen.

Anschließend werden neue NumberFormatter-, CurrencyFormatter- und DateTimeFormatter-Objekte für das Gebietsschema erstellt. Nach dem Erstellen der einzelnen Formatter-Objekte wird die traceError()-Methode aufgerufen. Diese Methode zeigt in der Konsole Fehler- und Warnmeldungen an, wenn ein Problem mit dem angeforderten Gebietsschema vorliegt. (In einer echten Anwendung sollten Fehlermeldungen nicht nur angezeigt werden, sondern es sollten auch entsprechende Maßnahmen ergriffen werden.)

Nach dem Erstellen des CurrencyFormatter-Objekts stellt die setLocale()-Methode den ISO-Währungscode, das Währungssymbol und die fractionalDigits-Eigenschaften des Objekts auf vorab bestimmte Werte ein. (Diese Werte werden immer eingestellt, wenn der Benutzer eine neue Börse im Kombinationsfeld auswählt.)

Nach dem Erstellen des DateTimeFormatter-Objekts ruft die setLocale()-Methode auch ein Array der lokalisierten Abkürzungen für Monatsnamen ab.

Formatieren der Daten

Die formatierten Börsendaten werden in einem DataGrid-Objekt angezeigt. Die DataGrid-Spalten rufen jeweils eine Beschriftungsfunktion auf, die den Spaltenwert anhand des entsprechenden Formatter-Objekts formatiert.

In der Flash-Version werden die DataGrid-Spalten beispielsweise mit dem folgenden Code eingerichtet:

```
var col1:DataGridColumn = new DataGridColumn("ticker");
col1.headerText = "Company";
col1.sortOptions = Array.NUMERIC;
col1.width = 200;

var col2:DataGridColumn = new DataGridColumn("volume");
col2.headerText = "Volume";
col2.width = 120;
col2.cellRenderer = RightAlignedCell;
col2.labelFunction = displayVolume;

var col3:DataGridColumn = new DataGridColumn("price");
col3.headerText = "Price";
col3.width = 70;
col3.cellRenderer = RightAlignedCell;
col3.labelFunction = displayPrice;

var col4:DataGridColumn = new DataGridColumn("change");
col4.headerText = "Change";
col4.width = 120;
col4.cellRenderer = RightAlignedCell;
col4.labelFunction = displayPercent;
```

Die Flex-Version des Beispiels deklariert das DataGrid-Objekt in MXML. Sie definiert auch ähnliche Beschriftungsfunktionen für die einzelnen Spalten.

Die labelFunction-Eigenschaften verweisen auf die folgenden Funktionen, die Formatierungsmethoden der Localizer-Klasse aufrufen:

```
private function displayVolume(item:Object):String
{
    return localizer.formatNumber(item.volume, 0);
}

private function displayPercent(item:Object):String
{
    return localizer.formatPercent(item.change )  ;
}

  private function displayPrice(item:Object):String
{
    return localizer.formatCurrency(item.price);
}
```

Dann werden die entsprechenden Formatter-Objekte von den Localizer-Methoden eingerichtet und aufgerufen:

```
public function formatNumber(value:Number, fractionalDigits:int = 2):String
{
    nf.fractionalDigits = fractionalDigits;
    return nf.formatNumber(value);
}

public function formatPercent(value:Number, fractionalDigits:int = 2):String
{
    // HACK WARNING: The position of the percent sign, and whether a space belongs
    // between it and the number, are locale-sensitive decisions. For example,
    // in Turkish the positive format is %12 and the negative format is -%12.
    // Like most operating systems, flash.globalization classes do not currently
    // provide an API for percentage formatting.
    nf.fractionalDigits = fractionalDigits;
    return nf.formatNumber(value) + "%";
}

public function formatCurrency(value:Number):String
{
    return cf.format(value, symbolIsSafe);
}

public function formatDate(dateValue:Date):String
{
    return df.format(dateValue);
}
|
```

Kapitel 57: Lokalisieren von Anwendungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Bei der Lokalisierung werden Elemente zum Unterstützen mehrerer Gebietsschemas hinzugefügt. Ein Gebietsschema ist eine Kombination aus einem Sprach- und einem Ländercode. en_US bezieht sich beispielsweise auf die in den USA gesprochene englische Sprache und fr_FR auf die in Frankreich gesprochene französische Sprache. Um eine Anwendung für diese Gebietsschemas zu lokalisieren, müssen Sie zwei Elementsätze bereitstellen: einen für das Gebietsschema „en_US“ und einen für das Gebietsschema „fr_FR“.

Unterschiedliche Gebietsschemas können die gleiche Sprache verwenden. en_US und en_GB (Großbritannien) sind beispielsweise unterschiedliche Gebietsschemas. In diesem Fall verwenden beide Gebietsschemas die englische Sprache. Der Ländercode weist jedoch darauf hin, dass es sich um unterschiedliche Gebietsschemas handelt, die möglicherweise unterschiedliche Elemente verwenden. In einer Anwendung für das Gebietsschema „en_US“ lautet das Word für Farbe möglicherweise „color“, im Gebietsschema „en_GB“ dagegen „colour“. Außerdem würde als Währung je nach Gebietsschema Dollar bzw. Pfund verwendet. Es kann ferner sein, dass sich das Datums- und Uhrzeitformat voneinander unterschieden.

Sie können einen Elementesatz für eine Sprache bereitstellen, ohne einen Ländercode anzugeben. So können Sie z. B. en-Elemente für die englische Sprache und zusätzliche Elemente für das Gebietsschema „en_US“ bereitstellen, die speziell für US-Englisch gelten.

Das Lokalisieren geht über die Übersetzung der in der Anwendung verwendeten Strings hinaus. Sie kann ferner beliebige Arten von Elementen umfassen, z. B. Audiodateien, Bilder und Videos.

Auswählen eines Gebietsschemas

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit folgenden Methoden können Sie ermitteln, welches Gebietsschema Ihr Inhalt oder Ihre Anwendung verwendet:

- flash.globalization package – Verwenden Sie die Klassen mit Gebietsschemaerkennung im flash.globalization-Paket, um das Standardgebietsschema für einen Benutzer auf Grundlage des Betriebssystems und der Benutzervoreinstellungen abzurufen. Dies ist das bevorzugte Verfahren für Anwendungen, die in Flash Player 10.1 oder höher oder in Laufzeiten von AIR 2.0 oder höher ausgeführt werden. Weitere Informationen finden Sie unter „Bestimmen des Gebietsschemas“ auf Seite 1008.
- Benutzeraufforderung – Sie können die Anwendung in einem Standardgebietsschema starten und den Benutzer auffordern, das bevorzugte Gebietsschema auszuwählen.

- `(Nur AIR) Capabilities.languages` – Die `Capabilities.languages`-Eigenschaft listet ein Array von Sprachen auf, die gemäß Einstellung im Betriebssystem als vom Benutzer bevorzugte Sprachen verfügbar sind. Die Strings enthalten von RFC4646 (http://www.ietf.org/rfc/rfc4646.txt) definierte Sprach-Tags (und, sofern zutreffend, Skript- und Gebietsschemainformationen). Die Strings verwenden Bindestriche als Trennzeichen (z. B. `"en-US"` oder `"ja-JP"`). Der erste Eintrag im zurückgegebenen Array hat die gleiche primäre Sprachen-ID wie die language-Eigenschaft. Wenn beispielsweise `languages[0]` auf `"en-US"` gesetzt ist, ist die `language`-Eigenschaft auf `"en" gesetzt.` Wenn aber die language-Eigenschaft auf `"xu"` eingestellt ist (also auf eine unbekannte Sprache), dann lautet das erste Element im `languages`-Array anders.
- `Capabilities.language` – Die `Capabilities.language`-Eigenschaft stellt den Benutzeroberflächensprachcode des Betriebssystems bereit. Diese Eigenschaft ist jedoch auf 20 bekannte Sprachen beschränkt. In englischen Systemen gibt diese Eigenschaft außerdem nur den Sprachcode, nicht jedoch den Ländercode zurück. Aus diesen Gründen empfiehlt es sich, das erste Element des `Capabilities.languages`-Arrays zu verwenden.

Lokalisieren von Flex-Inhalten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Adobe Flex umfasst eine Architektur zum Lokalisieren von Flex-Inhalten. Diese Architektur enthält die Locale-, ResourceBundle- und ResourceManagerImpl-Klassen sowie die IResourceBundle- und IResourceManagerImpl-Schnittstellen.

Eine Flex-Lokalisierungsbibliothek mit Dienstklassen zum Sortieren von Gebietsschemas für Anwendungen steht unter Google Code (http://code.google.com/p/as3localelib/) zur Verfügung.

Verwandte Hilfethemen

http://code.google.com/p/as3localelib/

Lokalisieren von Flash-Inhalten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Adobe Flash Professional enthält eine Locale-Klasse in den ActionScript 3.0-Komponenten. Mit der Locale-Klasse können Sie steuern, wie eine SWF-Datei in mehrsprachigem Text angezeigt wird. Das Flash-Bedienfeld „Strings“ gestattet die Verwendung von String-IDs in dynamischen Textfeldern anstelle von String-Literalen. Dadurch können Sie eine SWF-Datei erstellen, die Text anzeigt, der aus einer sprachspezifischen XML-Datei geladen wurde. Informationen zur Verwendung der Locale-Klasse finden Sie im ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform.

Lokalisieren von AIR-Anwendungen

Adobe AIR 1.0 und höher

Das AIR-SDK bietet (in der Datei „AIRLocalizer.js“) eine HTML-Lokalisierungsarchitektur. Die Architektur umfasst APIs für die Arbeit mit mehreren Gebietsschemas in einer HTML-Anwendung. Eine ActionScript-Bibliothek für das Sortieren von Gebietsschemas finden Sie unter http://code.google.com/p/as3localelib/.

Verwandte Hilfethemen

http://code.google.com/p/as3localelib/

Lokalisieren von Datum, Uhrzeit und Währungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Wiedergabe von Datum, Uhrzeit und Währungen in einer Anwendung hängt vom Gebietsschema ab. In den USA wird das Datum z. B. standardmäßig als Monat/Tag/Jahr, in Europa dagegen als Tag/Monat/Jahr wiedergegeben.

Sie können Code zum Formatieren von Datum, Uhrzeit und Währungen schreiben. Der folgende Code konvertiert ein Date-Objekt beispielsweise in das Format „Monat/Tag/Jahr“ bzw. „Tag/Monat/Jahr“. Wenn für die `locale`-Variable (die das Gebietsschema repräsentiert) `"en_US"` festgelegt wurde, gibt die Funktion das Format „Monat/Tag/Jahr“ zurück. Im Beispiel wird ein Date-Objekt für alle anderen Gebietsschemas in das Format „Tag/Monat/Jahr“ konvertiert:

```
function convertDate(date)
{
    if (locale == "en_US")
    {
        return (date.getMonth() + 1) + "/" + date.getDate() + "/" + date.getFullYear();
    }
    else
    {
        return date.getDate() + "/" + (date.getMonth() + 1) + "/" + date.getFullYear();
    }
}
```

ADOBE FLEX

Die Flex-Architektur umfasst Steuerelemente zum Formatieren von Datum, Uhrzeit und Währungen, darunter DateFormatter und CurrencyFormatter.

- mx:DateFormatter
- mx:CurrencyFormatter

Kapitel 58: Einführung in die HTML-Umgebung

Adobe AIR 1.0 und höher

Adobe® AIR® verwendet das auch vom Safari-Webbrowser eingesetzte WebKit *(www.webkit.org)* für die Analyse, das Layout und die Wiedergabe von HTML- und JavaScript-Inhalten. Die Verwendung von AIR-APIs in HTML-Inhalten ist optional. Die Inhalte eines HTMLLoader-Objekts oder HTML-Fensters können vollständig mit HTML und JavaScript programmiert werden. Bei den meisten bestehenden HTML-Seiten und -Anwendungen sind nur wenige Änderungen erforderlich, vorausgesetzt, die in ihnen verwendeten HTML-, CSS-, DOM- und JavaScript-Funktionen sind mit WebKit kompatibel.

Wichtig: Neue Versionen der Adobe AIR-Laufzeitumgebung enthalten ggf. aktualisierte Versionen des WebKit. Ein WebKit-Update in einer neuen AIR-Version *kann* zu unerwarteten Änderungen in einer bereitgestellten AIR-Anwendung führen. Diese Änderungen können sich auf das Verhalten oder Aussehen von HTML-Inhalten in einer Anwendung auswirken. Verbesserungen oder Korrekturen am WebKit-Rendering können zum Beispiel das Layout von Elementen in der Benutzeroberfläche einer Anwendung ändern. Aus diesem Grund wird dringend empfohlen, einen Updatemechanismus in Ihre Anwendungen zu integrieren. Sollte es nötig sein, Ihre Anwendung aufgrund einer Änderung an der in AIR enthaltenen WebKit-Version zu aktualisieren, kann der AIR-Updatemechanismus den Benutzer auffordern, die neue Version Ihrer Anwendung zu installieren.

Der folgenden Tabelle können Sie entnehmen, welche Version des Safari-Webbrowsers die WebKit-Version verwendet, die der in AIR verwendeten entspricht:

AIR-Version	Safari-Version
1.0	2.04
1.1	3.04
1.5	4.0 Beta
2.0	4.03
2.5	4.03
2.6	4.03
2.7	4.03
3	5.0.3

Sie können die installierte WebKit-Version jederzeit anhand des Standardbenutzer-Agentenstrings ermitteln, der von einem HTMLLoader-Objekt zurückgegeben wird:

```
var htmlLoader:HTMLLoader = new HTMLLoader();
trace( htmlLoader.userAgent );
```

Beachten Sie, dass die in AIR verwendete WebKit-Version nicht mit der Open-Source-Version identisch ist. Einige Funktionsmerkmale werden in AIR nicht unterstützt. Zudem kann die AIR-Version Sicherheitspatches und Korrekturen enthalten, die in der entsprechenden WebKit-Version noch nicht zur Verfügung stehen. Siehe „In AIR nicht unterstützte WebKit-Funktionsmerkmale“ auf Seite 1042.

Da AIR-Anwendungen direkt auf dem Desktop ausgeführt werden und über vollständigen Zugriff auf das Dateisystem verfügen, wird für die HTML-Inhalte ein enger gefasstes Sicherheitsmodell verwendet, als dies gemeinhin bei Webbrowsern der Fall ist. In AIR werden nur Inhalte, die aus dem Installationsverzeichnis der Anwendung geladen werden, in der *Anwendungs-Sandbox* platziert. Diese Anwendungs-Sandbox ist mit der höchsten Zugriffsebene versehen und gestattet den Zugriff auf die AIR-APIs. Andere Inhalte werden von AIR je nach Herkunft in allein stehende Sandboxen gestellt. Aus dem Dateisystem geladene Dateien werden in einer lokalen Sandbox untergebracht. Dateien, die mithilfe der Protokolle http: oder https: vom Netzwerk geladen werden, werden je nach Domäne des Remote-Servers in entsprechenden Sandboxen gespeichert. Die Inhalte in diesen anwendungsfremden Sandboxen können nicht auf AIR-APIs zugreifen und werden ähnlich wie Inhalte in gängigen Webbrowsern ausgeführt.

HTML-Inhalte in AIR zeigen keine SWF- oder PDF-Inhalte an, wenn die Einstellungen für Alpha, Skalierung oder Transparenz angewendet wurden. Weitere Informationen finden Sie unter „Erwägungen beim Laden von SWF- oder PDF-Inhalt in eine HTML-Seite“ auf Seite 1074 und „Fenstertransparenz“ auf Seite 952.

Verwandte Hilfethemen

WebKit-DOM-Referenz

Safari-HTML-Referenz

Safari-CSS-Referenz

www.webkit.org

Überblick über die HTML-Umgebung

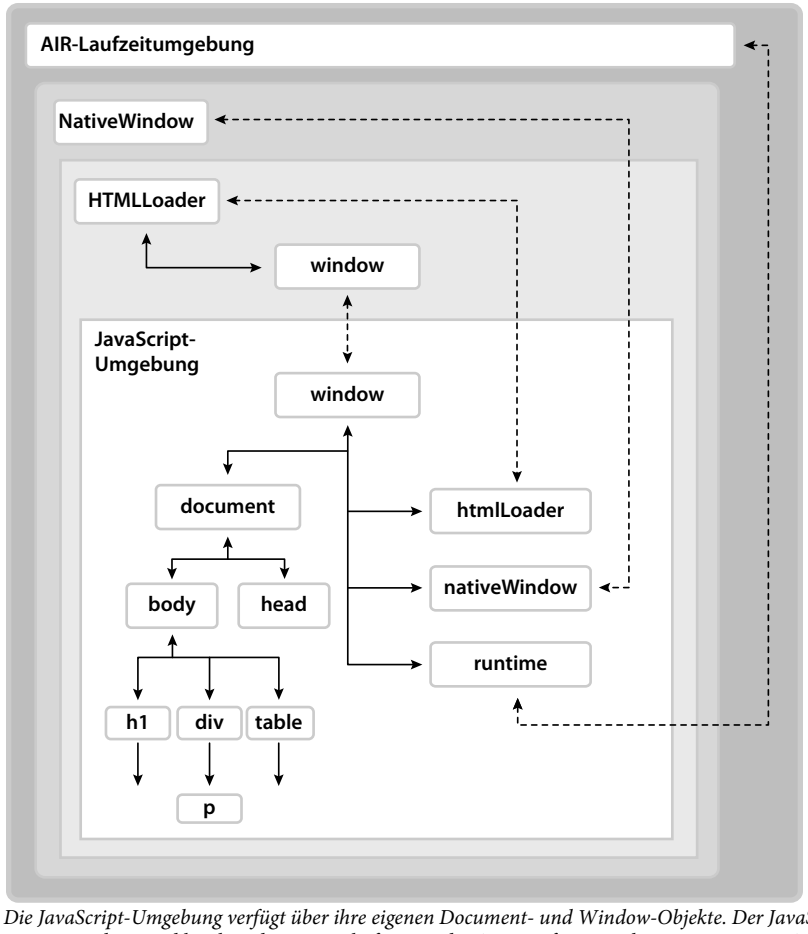
Adobe AIR 1.0 und höher

Adobe AIR bietet eine vollständige, browserähnliche JavaScript-Umgebung mit HTML-Renderer, Dokument-Objekt-Modell und JavaScript-Interpreter. Die JavaScript-Umgebung wird durch die HTMLLoader-Klasse in AIR dargestellt. In HTML-Fenstern enthält ein HTMLLoader-Objekt den gesamten HTML-Inhalt. Das HTMLLoader-Objekt wiederum befindet sich in einem NativeWindow-Objekt. In SWF-Inhalten kann die HTMLLoader-Klasse, die die Sprite-Klasse erweitert, wie jedes andere Anzeigeobjekt zur Anzeigeliste einer Bühne hinzugefügt werden. Die Adobe® ActionScript® 3.0-Eigenschaften der Klasse sind unter „Skripterstellung von AIR-HTML-Containern“ auf Seite 1072 beschrieben und außerdem auch im *ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform*. Im Flex-Framework legt sich eine mx:HTML-Komponente um die HTMLLoader-Klasse von AIR. Die mx:HTML-Komponente erweitert die UIComponent-Klasse, sodass diese direkt mit anderen Flex-Containern verwendet werden kann. Abgesehen davon ist die JavaScript-Umgebung innerhalb der mx:HTML-Komponente identisch.

JavaScript-Umgebung und der AIR-Host

Adobe AIR 1.0 und höher

Das folgende Diagramm verdeutlicht das Verhältnis zwischen der JavaScript-Umgebung und der AIR-Laufzeitumgebung. Es wird nur ein einzelnes natives Fenster dargestellt, eine AIR-Anwendung kann jedoch mehrere Fenster umfassen. (Einzelne Fenster können zudem mehrere HTMLLoader-Objekte enthalten.)



Die JavaScript-Umgebung verfügt über ihre eigenen Document- und Window-Objekte. Der JavaScript-Code kann über die Laufzeit-, NativeWindow- und htmlLoader-Eigenschaften mit der AIR-Laufzeitumgebung interagieren. ActionScript-Code kann mit der JavaScript-Umgebung über die window-Eigenschaft eines HTMLLoader-Objekts, das auf ein Window-Objekt von JavaScript verweist, interagieren. Zudem können sowohl ActionScript- als auch JavaScript-Objekte als Listener für AIR- oder JavaScript-Ereignisse verwendet werden.

Die Eigenschaft `runtime` ermöglicht den Zugriff auf AIR-API-Klassen und gibt Ihnen damit die Möglichkeit, neue AIR-Objekte zu erstellen und auf Klassenmitglieder (auch als statische Mitglieder bezeichnet) zuzugreifen. Um auf eine AIR-API zuzugreifen, fügen Sie den Namen der Klasse zusammen mit dem Packet zur Eigenschaft `runtime` hinzu. Zur Erstellung eines File-Objekts geben Sie zum Beispiel die folgende Anweisung ein:

```
var file = new window.runtime.filesystem.File();
```

Hinweis: *Das AIR-SDK stellt eine JavaScript-Datei (`AIRAliases.js`) zur Verfügung, in der geeignete Aliasnamen für die am häufigsten verwendeten AIR-Klassen definiert werden. Wenn Sie diese Datei importieren, können Sie anstatt des umständlichen „window.runtime.package.Class“ das einfachere „air.Class“ verwenden. So können Sie beispielsweise mit `new air.File()` das File-Objekt erstellen.*

Das NativeWindow-Objekt enthält Eigenschaften zur Steuerung des Desktop-Fensters. Auf dieses NativeWindow-Objekt können Sie mit der Eigenschaft `window.nativeWindow` aus einer HTML-Seite heraus zugreifen.

Das HTMLLoader-Objekt stellt Eigenschaften, Methoden und Ereignisse bereit, mit denen das Laden und die Wiedergabe von Inhalten gesteuert wird. Auf das übergeordnete HTMLLoader-Objekt können Sie mit der Eigenschaft `window.htmlLoader` aus einer HTML-Seite heraus zugreifen.

Wichtig: *Lediglich Seiten, die als Teil einer Anwendung installiert wurden, verfügen über die Eigenschaften* `htmlLoader`, `nativeWindow` *oder* `runtime` *und dies nur, wenn sie als Dokument der obersten Ebene geladen wurden. Die Eigenschaften werden nicht hinzugefügt, wenn das Dokument in einen Frame oder iFrame geladen wird. (Ein untergeordnetes Dokument kann auf diese Eigenschaften des übergeordneten Dokuments zugreifen, vorausgesetzt, es befindet sich in derselben Sicherheits-Sandbox. Ein Dokument, das in einen Frame geladen wurde, kann zum Beispiel mit* `parent.runtime` *auf die Eigenschaft* `runtime` *des übergeordneten Dokuments zugreifen.)*

Sicherheit

Adobe AIR 1.0 und höher

AIR führt den gesamten Code innerhalb einer Sicherheits-Sandbox, basierend auf der Ursprungsdomäne, aus. Anwendungsinhalte, also Inhalte, die aus dem Installationsverzeichnis der Anwendung geladen werden, werden in der *Anwendungs*-Sandbox platziert. Auf die Laufzeitumgebung und die AIR-APIs kann nur von HTML und JavaScript zugegriffen werden, das innerhalb dieser Sandbox ausgeführt wird. Gleichzeitig wird die dynamische Auswertung und Ausführung von JavaScript in der Anwendungs-Sandbox größtenteils blockiert, nachdem alle Prozeduren für das Ereignis `load` für die Seite zurückgegeben wurden.

Sie können eine Anwendungsseite einer anwendungsfremden Sandbox zuordnen, indem Sie die Seite in einen Frame oder iFrame laden und die AIR-spezifischen Attribute `sandboxRoot` und `documentRoot` des Frames einstellen. Durch die Einstellung des Werts `sandboxRoot` auf eine vorhandene Remote-Domäne können Skripts in der Sandbox auf Inhalte in dieser Domäne Bezug nehmen. Diese Art der Zuordnung von Seiten kann besonders beim Laden und Skripten von Remote-Inhalten wie in *mash-up*-Anwendungen nützlich sein.

Eine weitere Möglichkeit, gegenseitige Verweise in Skripten der Anwendung und anwendungsfremden Inhalten zuzulassen und das einzige Verfahren, mit dem anwendungsfremden Inhalten Zugriff auf AIR-APIs gewährt werden kann, ist die Erstellung einer *Sandbox-Brücke*. Eine *Brücke von übergeordneten zu untergeordneten Inhalten* ermöglicht Inhalten in untergeordneten Frames, iFrames oder Fenstern den Zugriff auf in der Anwendungs-Sandbox festgelegte Methoden und Eigenschaften. Eine *Brücke von untergeordneten zu übergeordneten Inhalten* ermöglicht wiederum den Zugriff von Anwendungsinhalten auf bestimmte Methoden und Eigenschaften, die in der Sandbox der untergeordneten Inhalte festgelegt wurden. Sandbox-Brücken werden durch die Eigenschaften `parentSandboxBridge` und `childSandboxBridge` im window-Objekt erstellt. Weitere Informationen finden Sie unter „HTML-Sicherheit in Adobe AIR“ auf Seite 1158 und „HTML-Frame- und iFrame-Elemente“ auf Seite 1038.

Zusatzmodule und eingebettete Objekte

Adobe AIR 1.0 und höher

AIR unterstützt das Adobe® Acrobat®-Zusatzmodul. Für die Anzeige von PDF-Inhalten muss auf dem Gerät Acrobat oder Adobe® Reader® 8.1 (oder höher) installiert sein. Das HTMLLoader-Objekt verfügt über eine Eigenschaft, mit der überprüft werden kann, ob auf dem System des Benutzers PDF-Dateien angezeigt werden können. SWF-Dateiinhalte können auch in der HTML-Umgebung angezeigt werden, dies ist jedoch eine Funktion von AIR, für die keine externen Zusatzmodule benötigt werden.

AIR unterstützt keine weiteren WebKit-Zusatzmodule.

Verwandte Hilfethemen

„HTML-Sicherheit in Adobe AIR“ auf Seite 1158

„HTML-Sandboxen“ auf Seite 1030

„HTML-Frame- und iFrame-Elemente“ auf Seite 1038

„JavaScript-Window-Objekt“ auf Seite 1036

„Das XMLHttpRequest-Objekt“ auf Seite 1032

„Hinzufügen von PDF-Inhalten in AIR“ auf Seite 587

AIR und WebKit

Adobe AIR 1.0 und höher

Adobe AIR verwendet die Open-Source-Engine WebKit, die auch im Webbrowser Safari Anwendung findet. Aus Sicherheitsgründen und um Zugriff auf die Laufzeitklassen und -objekte zu gewähren, bietet AIR mehrere Erweiterungen. WebKit selbst fügt zusätzliche Funktionen hinzu, die nicht in den W3C-Standards für HTML, CSS und JavaScript enthalten sind.

In den folgenden Abschnitten wird nur auf die Erweiterungen von AIR und die wichtigsten WebKit-Erweiterungen eingegangen. Weitere Hinweise zu vom Standard abweichendem HTML-, CSS- und JavaScript-Code finden Sie unter www.webkit.org und developer.apple.com. Informationen zu den Standards finden Sie auf derW3C-Website. Mozilla bietet außerdem wertvolle allgemeine Informationen zu HTML-, CSS- und DOM-Themen (wenn die WebKit- und Mozilla-Engine auch natürlich nicht identisch sind).

JavaScript in AIR

Flash Player 9 und höher, Adobe AIR 1.0 und höher

AIR verändert das typische Verhalten gängiger JavaScript-Objekte. Viele dieser Änderungen wurden vorgenommen, um die Erstellung sicherer Anwendungen in AIR zu erleichtern. Diese Veränderungen bedeuten jedoch auch, dass einige gängige JavaScript-Kodierungsmuster und vorhandene Internetanwendungen, die diese Muster einsetzen, nicht immer wie erwartet in AIR ausgeführt werden. Informationen zur Behebung solcher Probleme finden Sie unter „Vermeiden von sicherheitsbezogenen JavaScript-Fehlern“ auf Seite 1049.

HTML-Sandboxen

Adobe AIR 1.0 und höher

AIR platziert Inhalte je nach Herkunft in separaten Sandboxen. Die Sandbox-Regeln entsprechen sowohl den Herkunftskriterien der meisten Webbrowser als auch den Sandbox-Regeln von Adobe Flash Player. Zusätzlich stellt AIR einen neuen *Anwendungs*-Sandboxtyp zur Verfügung, der die Anwendungsinhalte enthält und schützt. Weitere Informationen zu den Sandboxtypen, die Ihnen bei der Entwicklung von AIR-Anwendungen begegnen können, finden Sie unter „Sicherheits-Sandboxen“ auf Seite 1117.

Auf die Laufzeitumgebung und die AIR-APIs kann nur von HTML und JavaScript zugegriffen werden, das innerhalb der Anwendungs-Sandbox ausgeführt wird. Gleichzeitig wird die dynamische Auswertung und Ausführung von JavaScript in seinen verschiedenen Ausformungen aus Sicherheitsgründen in weiten Teilen auf die Anwendungs-Sandboxen beschränkt. Diese Einschränkungen gelten unabhängig davon, ob Ihre Anwendung Informationen tatsächlich direkt von einem Server lädt. (Selbst Dateiinhalte, analysierte Strings und direkte Benutzereingaben sind nicht immer vertrauenswürdig.)

In welcher Sandbox eine Seite abgelegt wird, wird von dem Ursprung ihrer Inhalte bestimmt. Lediglich Inhalte, die aus dem Anwendungsverzeichnis (dem Installationsverzeichnis, auf das das URL-Schema `app:` verweist) geladen werden, werden in der Anwendungs-Sandbox platziert. Inhalte, die aus dem Dateisystem geladen werden, werden in der *local-with-filesystem-Sandbox* oder der *local-trusted-Sandbox* abgelegt, die den Zugriff auf und die Interaktion mit Inhalten des lokalen Dateisystems, jedoch nicht mit Remote-Inhalten zulassen. Vom Netzwerk geladene Inhalte werden in einer Remote-Sandbox platziert, die der Ursprungsdomäne der Inhalte entspricht.

Soll eine Anwendungsseite ungehindert mit Inhalten in einer Remote-Sandbox interagieren, kann die Seite derselben Domäne wie die Remote-Inhalte zugeordnet werden. Wenn Sie zum Beispiel eine Anwendung erstellen, die Kartendaten von einem Internet-Dienst anzeigt, können Sie die Seite der Anwendung, welche die Inhalte des Dienstes lädt und anzeigt, der Domäne dieses Dienstes zuordnen. Die Attribute für die Zuordnung von Seiten zu Remote-Sandboxen und -Domänen sind neue Attribute, die zu den frame- und iframe-HTML-Elementen hinzugefügt wurden.

Um sicherzustellen, dass Inhalte in einer anwendungsfremden Sandbox risikolos die AIR-Funktionen verwenden können, können Sie eine übergeordnete Sandbox-Brücke einrichten. Untergeordnete Sandbox-Brücken können eingerichtet werden, um zu gewährleisten, dass Anwendungsinhalte sicher Methoden aufrufen und auf Eigenschaften zugreifen können, die zu Inhalten anderer Sandboxen gehören. Sicherheit bedeutet hier, dass die Remote-Inhalte nicht versehentlich Verweise auf Objekte, Eigenschaften oder Methoden erhalten können, die nicht explizit bereitgestellt wurden. Es können nur einfache Datentypen, Funktionen und anonyme Objekte über diese Brücke übergeben werden. Sie müssen jedoch weiterhin dafür Sorge tragen, dass möglicherweise gefährliche Funktionen nicht explizit bereitgestellt werden. Sollten Sie zum Beispiel eine Schnittstelle bereitstellen, die es Remote-Inhalten ermöglichte, Dateien auf dem gesamten System eines Benutzers zu lesen und zu speichern, geben Sie den Remote-Inhalten damit die Chance, Ihren Benutzern erheblichen Schaden zuzufügen.

JavaScript-Funktion eval()

Adobe AIR 1.0 und höher

Sobald der Ladevorgang für eine Seite abgeschlossen ist, bleibt die Verwendung der Funktion `eval()` auf die Anwendungs-Sandbox beschränkt. Einige Verwendungsmöglichkeiten sind gestattet, damit JSON-formatierte Daten sicher analysiert werden können; Auswertungen, die zu ausführbaren Anweisungen führen, verursachen jedoch eine Fehlermeldung. Unter „Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen“ auf Seite 1161 wird beschrieben, welche Verwendungsmöglichkeiten für die Funktion `eval()` gestattet sind.

Function-Konstruktor

Adobe AIR 1.0 und höher

Function-Konstruktoren können in den Anwendungs-Sandboxen verwendet werden, bevor der Ladevorgang der Seite abgeschlossen ist. Wenn alle `load`-Ereignisprozeduren abgeschlossen sind, können keine neuen Funktionen erstellt werden.

Laden externer Skripte

Adobe AIR 1.0 und höher

HTML-Seiten in der Anwendungs-Sandbox können den `script`-Tag nicht für das Laden von JavaScript-Dateien verwenden, die sich außerhalb des Anwendungsverzeichnisses befinden. Soll eine Seite in Ihrer Anwendung ein Skript laden, das sich nicht im Anwendungsverzeichnis befindet, muss die Seite einer anwendungsfremden Sandbox zugeordnet werden.

Das XMLHttpRequest-Objekt

Adobe AIR 1.0 und höher

AIR stellt ein XMLHttpRequest-Objekt (XHR-Objekt) zur Verfügung, mit dem Anwendungen Daten anfordern können. Das folgende Beispiel zeigt eine einfache Datenanforderung:

```
xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", "http:/www.example.com/file.data", true);
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4) {
        //do something with data...
    }
}
xmlhttp.send(null);
```

Im Gegensatz zu Browsern lässt AIR Inhalte, die in der Anwendungs-Sandbox ausgeführt werden, Daten von beliebigen Domänen anfordern. Das Ergebnis eines XHR, das einen JSON-String enthält, kann in Datenobjekte ausgewertet werden, sofern das Ergebnis keinen ausführbaren Code umfasst. Befinden sich im XHR-Ergebnis ausführbare Anweisungen, wird ein Fehler ausgegeben und der Auswertungsversuch schlägt fehl.

Um zu verhindern, das versehentlich Code aus Remote-Quellen eingefügt wird, geben synchrone XHRs ein leeres Ergebnis zurück, wenn die Anforderungen durchgeführt wurden, bevor der Ladevorgang der Seite abgeschlossen wurde. Asynchrone XHRs geben das Ergebnis immer nach dem Laden der Seite zurück.

Domänenübergreifende XMLHttpRequests in anwendungsfremden Sandboxen werden von AIR standardmäßig blockiert. Übergeordnete Fenster in der Anwendungs-Sandbox können wählen, ob domänenübergreifende Anforderungen in einem untergeordnetem Frame, der Inhalte in einer anwendungsfremden Sandbox enthält, zugelassen werden. Hierfür wird `allowCrossDomainXHR`, ein von AIR hinzugefügtes Attribut im Container-frame oder -iframe-Element auf `true` gesetzt:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    allowCrossDomainXHR="true"
</iframe>
```

Hinweis: *Bei Bedarf kann die URLStream-Klasse von AIR auch für das Herunterladen von Daten eingesetzt werden.*

Wenn Sie von einem Frame oder iFrame, der einer Remote-Sandbox zugeordnete Anwendungsinhalte umfasst, eine XMLHttpRequest-Anforderung an einen Remote-Server auslösen, müssen Sie sicherstellen, dass die Zuordnungs-URL nicht die in der XHR verwendete Serveradresse maskiert. Das folgende Beispiel zeigt eine iframe-Definition, mit der Anwendungsinhalte einer Remote-Sandbox für die Domäne „example.com“ zugeordnet werden:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/"
    allowCrossDomainXHR="true"
</iframe>
```

Da das Attribut `sandboxRoot` die Stamm-URL der Adresse www.example.com neu zuordnet, werden alle Anforderungen aus dem Anwendungsverzeichnis und nicht vom Remote-Server geladen. Anforderungen werden unabhängig davon, ob sie aus einer Seitennavigation oder einer XMLHttpRequest-Anforderung hervorgehen, neu zugeordnet.

Um zu vermeiden, dass Datenanforderungen an den Remote-Server versehentlich blockiert werden, ordnen Sie `sandboxRoot` nicht der Stamm-URL sondern einem Unterverzeichnis der Remote-URL zu. Dieses Verzeichnis muss nicht tatsächlich vorhanden sein. Um Anforderungen an „www.example.com“ für das Laden vom Remote-Server anstelle des Anwendungsverzeichnisses zu gestatten, ändern Sie den iFrame wie folgt:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/air/"
    allowCrossDomainXHR="true"
</iframe>
```

In diesem Fall werden lediglich Inhalte im Unterverzeichnis `air` lokal geladen.

Weitere Informationen zur Sandboxen-Zuordnung finden Sie unter „HTML-Frame- und iFrame-Elemente“ auf Seite 1038 und „HTML-Sicherheit in Adobe AIR“ auf Seite 1158.

Cookies

Adobe AIR 1.0 und höher

In AIR-Anwendungen können nur Inhalte in Remote-Sandboxen (aus http:- und https:-Quellen geladene Inhalte) Cookies verwenden (die Eigenschaft `document.cookie`). In der Anwendungs-Sandbox stehen andere Mittel zum Speichern bleibender Daten zur Verfügung, darunter die EncryptedLocalStore-, SharedObject- und FileStream-Klassen.

Das Clipboard-Objekt

Adobe AIR 1.0 und höher

Die Clipboard-API von Webkit wird durch die folgenden Ereignisse gesteuert: `copy`, `cut` und `paste`. Das in diesen Ereignissen übergebene event-Objekt bietet mithilfe der Eigenschaft `clipboardData` Zugriff auf die Zwischenablage. Verwenden Sie die folgenden Methoden des Objekts `clipboardData`, um Daten aus der Zwischenablage zu lesen oder in diese zu schreiben:

Methode	Beschreibung
clearData(mimeType)	Löscht die Daten in der Zwischenablage. Setzen Sie den Parameter für `mimeType` auf den MIME-Typ der zu löschenden Daten.
getData(mimeType)	Ruft die Daten aus der Zwischenablage auf. Diese Methode kann nur in einer Prozedur für das Ereignis `paste` aufgerufen werden. Setzen Sie den Parameter für `mimeType` auf den MIME-Typ der zurückzugebenden Daten.
setData(mimeType, Daten)	Kopiert Daten in die Zwischenablage. Setzen Sie den Parameter für `mimeType` auf den MIME-Typ der Daten.

JavaScript-Code außerhalb der Anwendungs-Sandbox kann nur durch diese Ereignisse auf die Zwischenablage zugreifen. Die Inhalte in der Anwendungs-Sandbox können jedoch mithilfe der Clipboard-Klasse von AIR direkt auf die Zwischenablage des Systems zugreifen. Verwenden Sie zum Beispiel die folgende Anweisung, um Textformatdaten in die Zwischenablage zu kopieren:

```
var clipping = air.Clipboard.generalClipboard.getData("text/plain",
                                air.ClipboardTransferMode.ORIGINAL_ONLY);
```

Folgende MIME-Typen sind für die Daten gültig:

MIME-Typ	Wert
Text	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
File list	"application/x-vnd.adobe.air.file-list"

Wichtig: *Nur die Inhalte in der Anwendungs-Sandbox können auf Dateidaten in der Zwischenablage zugreifen. Unternehmen anwendungsfremde Inhalte den Versuch, auf ein Dateiobjekt in der Zwischenablage zuzugreifen, wird ein Sicherheitsfehler ausgelöst.*

Weitere Informationen zur Verwendung der Zwischenablage finden Sie unter „Kopieren und Einfügen“ auf Seite 635 und Using the Pasteboard from JavaScript (Apple Developer Center).

Drag & Drop (Ziehen und Ablegen)

Adobe AIR 1.0 und höher

Durch Drag & Drop in und aus HTML werden folgende DOM-Ereignisse verursacht: `dragstart`, `drag`, `dragend`, `dragenter`, `dragover`, `dragleave` und `drop`. Das in diesen Ereignissen übergebene event-Objekt bietet mithilfe der Eigenschaft `dataTransfer` Zugriff auf die gezogenen Daten. Die Eigenschaft `dataTransfer` verweist auf ein Objekt, das dieselben Methoden bereitstellt wie das mit einem Clipboard-Ereignis verknüpfte Objekt `clipboardData`. Sie können zum Beispiel folgende Funktion verwenden, um Textformatdaten aus einem `drop`-Ereignis zu erhalten.

```
function onDrop(dragEvent){
    return dragEvent.dataTransfer.getData("text/plain",
            air.ClipboardTransferMode.ORIGINAL_ONLY);
}
```

Das Objekt `dataTransfer` verfügt über folgende wichtige Mitglieder:

Mitglied	Beschreibung
clearData(mimeType)	Löscht die Daten. Setzen Sie den Parameter für `mimeType` auf den MIME-Typ der zu löschenden Datendarstellung.
getData(mimeType)	Ruft die gezogenen Daten auf. Diese Methode kann nur in einer Prozedur für das Ereignis `drop` aufgerufen werden. Setzen Sie den Parameter für `mimeType` auf den MIME-Typ der abzurufenden Daten.
setData(mimeType, Daten)	Geben Sie die zu ziehenden Daten an: Setzen Sie den Parameter für `mimeType` auf den MIME-Typ der Daten.
types	Ein Array von Strings, welches die MIME-Typen aller Datendarstellungen enthält, die im Objekt `dataTransfer` gegenwärtig zur Verfügung stehen.
effectsAllowed	Gibt an, ob die zu ziehenden Daten kopiert, verschoben oder verknüpft werden sollen oder eine Kombination dieser Vorgänge ausgeführt werden soll. Setzen Sie die Eigenschaft `effectsAllowed` in der Prozedur für das Ereignis `dragstart`.
dropEffect	Gibt an, welche der möglichen Effekte beim Ablegen von dem Ziel, auf das gezogen wird, unterstützt werden. Setzen Sie die Eigenschaft `dropEffect` in der Prozedur für das Ereignis `dragEnter`. Während der Ziehbewegung verwandelt sich der Cursor und zeigt so an, welchen Effekt ein Loslassen der Maustaste hat. Wurde kein `dropEffect` angegeben, wird ein Effekt der Eigenschaft `effectsAllowed` gewählt. Der Effekt „Kopieren“ hat Priorität vor dem Effekt „Verschieben“, der wiederum Priorität vor dem Effekt „Verknüpfen“ hat: Der Benutzer kann die Standardpriorität über die Tastatur ändern.

Weitere Information zur Unterstützung für das Drag & Drop in AIR-Anwendungen finden Sie unter „Ziehen und Ablegen in AIR“ auf Seite 648 und Using the Drag-and-Drop from JavaScript (Apple Developer Center).

Die innerHTML- und outerHTML-Eigenschaften

Adobe AIR 1.0 und höher

AIR platziert Sicherheitseinschränkungen für die Verwendung der Eigenschaften `innerHTML` und `outerHTML` für Inhalte, die in der Anwendung ausgeführt werden. Vor dem Seitenladeereignis und während der Ausführung von Ladeereignisprozeduren können die Eigenschaften `innerHTML` und `outerHTML` uneingeschränkt verwendet werden. Sobald das Laden der Seite abgeschlossen ist, können die Eigenschaften `innerHTML` oder `outerHTML` jedoch nur noch verwendet werden, um statische Inhalte zum Dokument hinzuzufügen. Anweisungen in dem `innerHTML` oder `outerHTML` zugewiesenem String, die einen ausführbaren Code ergeben, werden ignoriert. Wenn Sie beispielsweise ein Attribut für den Ereignisrückruf in eine Elementdefinition aufnehmen, wird der Ereignis-Listener nicht hinzugefügt. Eingebettete `<script>`-Tags werden ebenfalls nicht ausgewertet. Weitere Informationen finden Sie unter „HTML-Sicherheit in Adobe AIR“ auf Seite 1158.

Die Document.write()- und Document.writeln()-Methoden

Adobe AIR 1.0 und höher

Die Verwendung der Methoden `write()` und `writeln()` ist in der Anwendungs-Sandbox vor dem Ereignis `load` der Seite nicht beschränkt. Nachdem die Seite geladen wurde, wird durch den Aufruf dieser Methoden die Seite jedoch weder gelöscht noch wird eine neue Seite erstellt. In einer anwendungsfremden Sandbox wird durch den Aufruf von `document.write()` oder `writeln()` nach dem Laden einer Seite die aktuelle Seite wie in anderen Webbrowsern auch gelöscht und es wird eine neue, leere Seite geöffnet.

Die Eigenschaft Document.designMode

Adobe AIR 1.0 und höher

Setzen Sie die Eigenschaft `document.designMode` auf den Wert `on`, um alle Elemente im Dokument editierbar zu machen. Die integrierte Editorunterstützung umfasst die Bearbeitung, das Kopieren und Einfügen und Drag & Drop von Text. Die Einstellung von `designMode` auf `on` hat die gleichen Auswirkungen wie die Einstellung des Elements `body` der Eigenschaft `contentEditable` auf `true`. Sie können die Eigenschaft `contentEditable` auf die meisten HTML-Elemente anwenden, um festzulegen, welche Abschnitte eines Dokuments editierbar sein sollen. Weitere Informationen finden Sie unter „HTML-Attribut contentEditable“ auf Seite 1041.

unload-Ereignisse (für body- und frameset-Objekte)

Adobe AIR 1.0 und höher

Verwenden Sie das Ereignis `unload` nicht im obersten `frameset`- oder `body`-Tag eines Fensters (einschließlich des Hauptfensters der Anwendung), um auf das Schließen des Fensters (oder der Anwendung) zu reagieren. Verwenden Sie stattdessen das `exiting`-Ereignis des NativeApplication-Objekts, um festzustellen, wann eine Anwendung geschlossen wird, oder das `closing`-Ereignis des NativeWindow-Objekts, um festzustellen, wann ein Fenster geschlossen wird. Mit dem folgenden JavaScript-Code wird zum Beispiel eine Meldung (`"Goodbye."`) eingeblendet, wenn der Benutzer die Anwendung schließt:

```
var app = air.NativeApplication.nativeApplication;
app.addEventListener(air.Event.EXITING, closeHandler);
function closeHandler(event)
{
    alert("Goodbye.");
}
```

Skripts *können* jedoch erfolgreich auf ein `unload`-Ereignis reagieren, das durch die Navigation von Frame-, iFrame- oder Fensterinhalten der obersten Ebene ausgelöst wird.

Hinweis: *Diese Einschränkungen werden möglicherweise in zukünftigen Versionen von Adobe AIR aufgehoben.*

JavaScript-Window-Objekt

Adobe AIR 1.0 und höher

Das Window-Objekt bleibt das globale Objekt im JavaScript-Ausführungskontext. In der Anwendungs-Sandbox fügt AIR neue Eigenschaften zum JavaScript-Window-Objekt hinzu, um Zugriff auf die integrierten Klassen von AIR sowie wichtige Host-Objekte zu gewähren. Einige Methoden und Eigenschaften verhalten sich außerdem unterschiedlich, je nachdem ob sie sich in der Anwendungs-Sandbox befinden oder nicht.

Window.runtime-Eigenschaft Die Eigenschaft `runtime` gibt Ihnen die Möglichkeit, die integrierten runtime-Klassen aus der Anwendungs-Sandbox heraus zu instanziieren und zu verwenden. Zu diesen Klassen gehören die AIR- und Flash Player-APIs (jedoch zum Beispiel nicht das Flex-Framework). Mit der folgenden Anweisung wird beispielsweise ein AIR-File-Objekt erstellt:

```
var preferencesFile = new window.runtime.flash.filesystem.File();
```

Die Datei `AIRAliases.js`, die mit der AIR-SDK bereitgestellt wird, enthält alle Aliasdefinitionen, mit denen Sie solche Verweise kürzen können. Wird die Datei `AIRAliases.js` zum Beispiel in eine Seite importiert, kann das File-Objekt mit der folgenden Anweisung erstellt werden.

```
var preferencesFile = new air.File();
```

Die Eigenschaft `window.runtime` wird nur für Inhalte in der Anwendungs-Sandbox definiert und dies nur für das übergeordnete Dokument einer Seite mit Frames oder iFrames.

Weitere Informationen finden Sie unter „Verwenden der AIRAliases.js-Datei“ auf Seite 1055.

Window.nativeWindow-Eigenschaft Die Eigenschaft `nativeWindow` stellt einen Verweis auf das darunterliegende native Window-Objekt zur Verfügung. Mit dieser Eigenschaft können Sie Window-Funktionen und -Eigenschaften wie die Fensterposition, das Fensterformat und die Sichtbarkeit skripten und Window-Ereignisse wie Schließen, Größenveränderungen und Verschieben verarbeiten. Mit der folgenden Anweisung wird das Fenster zum Beispiel geschlossen:

```
window.nativeWindow.close();
```

Hinweis: *Die vom NativeWindow-Objekt bereitgestellten Funktionen zur Fenstersteuerung überschneiden sich mit den Funktionen des Window-Objekts von JavaScript. In solchen Fällen können Sie die Methode verwenden, die Ihnen am besten liegt.*

Die Eigenschaft `window.nativeWindow` wird nur für Inhalte in der Anwendungs-Sandbox definiert und dies nur für das übergeordnete Dokument einer Seite mit Frames oder iFrames.

Window.htmlLoader-Eigenschaft Die Eigenschaft `htmlLoader` bietet einen Verweis auf das HTMLLoader-Objekt von AIR, das die HTML-Inhalte enthält. Mit dieser Eigenschaft können Sie das Aussehen und das Verhalten der HTML-Umgebung skripten. So können Sie die Eigenschaft `htmlLoader.paintsDefaultBackground` zum Beispiel verwenden, um festzulegen, ob durch die Steuerung ein standardmäßiger weißer Hintergrund erstellt wird:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Hinweis: *Das HTMLLoader-Objekt selbst verfügt über eine `window`-Eigenschaft, die auf das Window-Objekt von JavaScript der in ihm enthaltenen HTML-Inhalte verweist. Mit dieser Eigenschaft können Sie über einen Verweis auf das Container-HTMLLoader-Objekt auf die JavaScript-Umgebung zugreifen.*

Die Eigenschaft `window.htmlLoader` wird nur für Inhalte in der Anwendungs-Sandbox definiert und dies nur für das übergeordnete Dokument einer Seite mit Frames oder iFrames.

Window.parentSandboxBridge- und Window.childSandboxBridge-Eigenschaften Mit den Eigenschaften `parentSandboxBridge` und `childSandboxBridge` können Sie eine Schnittstelle zwischen einem übergeordneten und einem untergeordneten Frame erstellen. Weitere Informationen finden Sie unter „Cross-Scripting von Inhalten in unterschiedlichen Sicherheits-Sandboxen“ auf Seite 1067.

Window.setTimeout()- und Window.setInterval()-Funktionen AIR gibt Sicherheitseinschränkungen für die Verwendung der Funktionen von `setTimeout()` und `setInterval()` in der Anwendungs-Sandbox vor. Wenn Sie `setTimeout()` oder `setInterval()` aufrufen, können Sie den auszuführenden Code nicht als String definieren. Stattdessen müssen Sie eine Funktionsreferenz verwenden. Weitere Informationen finden Sie unter „setTimeout() und setInterval()“ auf Seite 1052.

Window.open()-Funktion Wenn die Methode `open()` von einem Code, der in einer anwendungsfremden Sandbox ausgeführt wird, aufgerufen wird, öffnet die Methode nur ein Fenster, wenn dieser Aufruf durch eine Benutzeraktion (wie einen Mausklick oder eine Tastatureingabe ausgelöst wurde. Außerdem wird vor dem Fenstertitel der Anwendungstitel angezeigt (um zu verhindern, dass Fenster, die durch Remote-Inhalte geöffnet werden, von der Anwendung geöffnete Fenster imitieren). Weitere Informationen finden Sie unter „Beschränkungen beim Aufrufen der JavaScript-window.open()-Methode“ auf Seite 1164.

air.NativeApplication-Objekt

Adobe AIR 1.0 und höher

Das Objekt NativeApplication hält Informationen zum Anwendungsstatus bereit, löst verschiedene wichtige Ereignisse auf Anwendungsebene aus und bietet nützliche Funktionen für die Steuerung des Anwendungsverhaltens. Eine einzelne Instanz des Objekts NativeApplication wird automatisch erstellt. Über die klassendefinierte Eigenschaft `NativeApplication.nativeApplication` kann darauf zugegriffen werden.

Mit dem folgenden JavaScript-Code können Sie zum Beispiel auf das Objekt zugreifen:

```
var app = window.runtime.flash.desktop.NativeApplication.nativeApplication;
```

Wurde das Skript `AIRAliases.js` importiert, können Sie die kürzere Form verwenden:

```
var app = air.NativeApplication.nativeApplication;
```

Auf das NativeApplication-Objekt kann nur innerhalb der Anwendungs-Sandbox zugegriffen werden. Weitere Informationen zum NativeApplication-Objekt finden Sie unter „Arbeiten mit AIR-Laufzeit- und Betriebssysteminformationen“ auf Seite 945.

Das JavaScript-URL-Schema

Adobe AIR 1.0 und höher

Die Ausführung von in einem JavaScript- URL-Schema definierten Code (z. B. `href="javascript:alert('Test')"`) wird innerhalb der Anwendungs-Sandbox blockiert. Es wird kein Fehler ausgegeben.

HTML in AIR

Adobe AIR 1.0 und höher

AIR und WebKit definieren einige vom Standard abweichende HTML-Elemente und -Attribute, darunter:

„HTML-Frame- und iFrame-Elemente“ auf Seite 1038

„HTML-Element-Ereignisprozeduren“ auf Seite 1040

HTML-Frame- und iFrame-Elemente

Adobe AIR 1.0 und höher

AIR fügt neue Attribute zu den frame- und iframe-Elementen von Inhalten in der Anwendungs-Sandbox hinzu:

sandboxRoot-Attribut Das Attribut `sandboxRoot` gibt eine alternative, anwendungsfremde Ursprungsdomäne für die vom frame-Attribut `src` angegebene Datei an. Die Datei wird in die der Domäne entsprechenden anwendungsfremden Sandbox geladen. Inhalte in der Datei und aus der angegebenen Domäne geladene Inhalte können in Skripten gegenseitig aufeinander Bezug nehmen.

Wichtig: *Wenn Sie den Wert von `sandboxRoot` auf die Basis-URL der Domäne setzen, werden alle Anforderungen nach Inhalten dieser Domäne aus dem Anwendungsverzeichnis anstelle des Remote-Servers geladen (unabhängig davon, ob diese Anforderung durch eine Seitennavigation, eine XMLHttpRequest oder ein anderes Verfahren für das Laden von Inhalten ausgelöst wurde).*

documentRoot-Attribut Das Attribut `documentRoot` gibt das lokale Verzeichnis an, von dem URLs geladen werden, die innerhalb des durch `sandboxRoot` angegebenen Verzeichnisses zu Dateien aufgelöst werden.

Bei der Auflösung von URLs im frame-Attribut `src` oder in in den Frame geladene Inhalte, wird der Teil der URL, der dem in `sandboxRoot` angegebenen Wert entspricht, durch den in `documentRoot` angegebenen Wert ersetzt. Somit wird mit dem folgenden frame-Tag:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

`child.html` aus dem Unterverzeichnis `sandbox` des Anwendungsinstallationsordners geladen. Relative URLs in `child.html` werden auf Grundlage des Verzeichnisses `sandbox` aufgelöst. Beachten Sie, dass auf Dateien auf dem Remote-Server unter `www.example.com/air` nicht im Frame zugegriffen werden kann, da AIR versuchen würde, die Dateien aus dem Verzeichnis „app:/sandbox/“ zu laden.

allowCrossDomainXHR-Attribut Nehmen Sie in das Tag des einleitendes Frames die Angabe `allowCrossDomainXHR="allowCrossDomainXHR"` auf, damit Inhalte im Frame XMLHttpRequests an Remote-Domänen stellen können. Anwendungsfremde Inhalte können solche Anforderungen standardmäßig nur an die eigene Ursprungsdomäne stellen. Eine Ausweitung von XHRs über Domänen hinweg stellt weitreichende Herausforderungen an die Sicherheit. Code auf der Seite kann Daten mit beliebigen Domänen austauschen. Wurden auf irgendeinem Wege schädliche Inhalte in die Seite eingefügt, können die Daten, auf die durch Code in der aktuellen Sandbox zugegriffen werden kann, beschädigt werden. Aktivieren Sie domänenübergreifende XHRS daher nur für Seiten, die Sie erstellen und steuern und nur dann, wenn das Laden von Daten über Domänen hinweg wirklich erforderlich ist. Validieren Sie zudem alle externen Daten, die von der Seite geladen werden, um zu verhindern, dass Code eingefügt wird oder andere Angriffe auf die Seite ausgeübt werden.

Wichtig: *Wurde das Attribut `allowCrossDomainXHR` in ein frame- oder iframe-Element aufgenommen, sind domänenübergreifende XHR-Anforderungen aktiviert (es sei denn, der zugewiesene Wert ist „0“ oder beginnt mit den Buchstaben „f“ oder „n“). Durch das Setzen von `allowCrossDomainXHR` auf „`deny`" wären domänenübergreifende XHR-Anforderungen zum Beispiel immer noch möglich. Nehmen Sie das Attribut nicht in die Element-Deklaration auf, wenn Sie keine domänenübergreifenden Anforderungen zulassen wollen.*

ondominitialize-Attribut Gibt die Prozedur für das Ereignis `dominitialize` eines Frames an. Dieses Ereignis ist ein AIR-spezifisches Ereignis, das ausgelöst wird, wenn die Window- und Document-Objekte des Frames erstellt wurden, jedoch noch keine Skripts übergeben oder Document-Elemente erstellt wurden.

Der Frame löst das Ereignis `dominitialize` früh genug während der Ladesequenz aus, sodass Skripts in der untergeordneten Seite auf Objekte, Variablen und Funktionen verweisen können, die durch die Prozedur `dominitialize` zum untergeordneten Dokument hinzugefügt wurden. Die übergeordnete Seite muss sich in derselben Sandbox wie die untergeordnete Seite befinden, damit Objekte in einem untergeordneten Dokument direkt hinzugefügt werden können oder auf sie zugegriffen werden kann. Eine übergeordnete Seite in der Anwendungs-Sandbox kann jedoch eine Sandbox-Brücke erstellen, um mit Inhalten in einer anwendungsfremden Sandbox zu kommunizieren.

Im folgenden Beispiel wird die Verwendung des iframe-Tags in AIR dargestellt:

Platzieren Sie `child.html` in einer Remote-Sandbox, ohne eine Zuordnung zu einer tatsächlichen Domäne auf einem Remote-Server zu erstellen:

```
<iframe src="http://localhost/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://localhost/air/"/>
```

Platzieren Sie `child.html` in einer Remote-Sandbox und gestatten Sie XMLHttpRequest-Anforderungen nur für `www.example.com`:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

Platzieren Sie `child.html` in einer Remote-Sandbox und gestatten Sie XMLHttpRequest-Anforderungen an beliebige Remote-Domänen:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"
            allowCrossDomainXHR="allowCrossDomainXHR"/>
```

Platzieren Sie `child.html` in einer local-with-filesystem-Sandbox:

```
<iframe     src="file:///templates/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="app-storage:/templates/"/>
```

Platzieren Sie `child.html` in einer Remote-Sandbox und verwenden Sie das Ereignis `dominitialize`, um eine Sandbox-Brücke zu erstellen:

```
<html>
<head>
<script>
var bridgeInterface = {};
bridgeInterface.testProperty = "Bridge engaged";
function engageBridge(){
    document.getElementById("sandbox").parentSandboxBridge = bridgeInterface;
}
</script>
</head>
<body>
<iframe id="sandbox"
            src="http://www.example.com/air/child.html"
            documentRoot="app:/"
            sandboxRoot="http://www.example.com/air/"
            ondominitialize="engageBridge()"/>
</body>
</html>
```

Das folgende `child.html`-Dokument verdeutlicht, wie untergeordnete Inhalte auf die übergeordnete Sandbox-Brücke zugreifen können:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Weitere Informationen finden Sie unter „Cross-Scripting von Inhalten in unterschiedlichen Sicherheits-Sandboxen“ auf Seite 1067 und „HTML-Sicherheit in Adobe AIR“ auf Seite 1158.

HTML-Element-Ereignisprozeduren

Adobe AIR 1.0 und höher

DOM-Objekte in AIR und WebKit lösen verschiedene Ereignisse aus, die im Standard-DOM-Ereignismodell nicht zur Verfügung stehen. In der folgenden Tabelle werden die damit zusammenhängenden Ereignisattribute aufgeführt, die Sie einsetzen können, um Prozeduren für diese Ereignisse festzulegen:

Name des Rückrufattributs	Beschreibung
oncontextmenu	Wird aufgerufen, wenn ein Kontextmenü durch eine Aktion wie einen rechten Mausklick auf dem ausgewählten Text, aufgerufen wird.
oncopy	Wird aufgerufen, wenn eine Auswahl in einem Element kopiert wird.
oncut	Wird aufgerufen, wenn eine Auswahl in einem Element ausgeschnitten wird.
ondominitialize	Wird aufgerufen, wenn das DOM eines in einen Frame oder iFrame geladenen Dokuments erstellt wird, jedoch bevor DOM-Elemente erstellt oder Skripts analysiert werden.
ondrag	Wird beim Ziehen eines Elements aufgerufen.

Name des Rückrufattributs	Beschreibung
ondragend	Wird aufgerufen, wenn das gezogene Element losgelassen wird.
ondragenter	Wird aufgerufen, wenn eine Ziehbewegung über die Begrenzung eines Elements geführt wird.
ondragleave	Wird aufgerufen, wenn eine Ziehbewegung die Begrenzung eines Elements verlässt.
ondragover	Wird wiederholt aufgerufen, während eine Ziehbewegung innerhalb der Begrenzung eines Elements geführt wird.
ondragstart	Wird zu Beginn einer Ziehbewegung aufgerufen.
ondrop	Wird aufgerufen, wenn eine Ziehbewegung über einem Element abgebrochen wird.
onerror	Wird aufgerufen, wenn während des Ladens eines Elements ein Fehler auftritt.
oninput	Wird aufgerufen, wenn Text in ein Formularelement eingegeben wird.
onpaste	Wird aufgerufen, wenn ein Eintrag in ein Element eingefügt wird.
onscroll	Wird aufgerufen, wenn durch die Inhalte eines scrollbaren Elements gescrollt wird.
onselectstart	Wird zu Beginn einer Auswahl aufgerufen.

HTML-Attribut contentEditable

Adobe AIR 1.0 und höher

Sie können das Attribut `contentEditable` zu jedem beliebigen HTML-Attribut hinzufügen, um Benutzern die Möglichkeit zu geben, die Inhalte des Elements zu bearbeiten. Mit dem folgenden HTML-Code wird beispielsweise das gesamte Dokument als editierbar definiert, mit Ausnahme des ersten `p`-Elements:

```
<html>
<head/>
<body contentEditable="true">
    <h1>de Finibus Bonorum et Malorum</h1>
    <p contentEditable="false">Sed ut perspiciatis unde omnis iste natus error.</p>
    <p>At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis.</p>
</body>
</html>
```

Hinweis: *Wenn Sie die Eigenschaft `document.designMode` auf `on` setzen, sind alle Elemente des Dokuments editierbar, unabhängig davon, ob für ein einzelnes Element `contentEditable` gesetzt wurde. Wird jedoch `designMode` auf `off` gesetzt, bleiben Elemente, für die `contentEditable` auf `true` gesetzt wurde, weiterhin editierbar. Weitere Informationen finden Sie unter „Die Eigenschaft Document.designMode“ auf Seite 1035.*

Data:-URLs

Adobe AIR 2 und höher

AIR unterstützt `data:`-URLs für die folgenden Elemente:

- img
- Eingabetyp=„image“

- CSS-Regeln, die Bilder (wie Hintergrundbilder) zulassen

Mithilfe von Data-URLs können Sie Binärdaten für Bilder als base64-kodierten String direkt in ein CSS- oder HTML-Dokument einfügen. Im folgenden Beispiel wird ein data:-URL für einen sich wiederholenden Hintergrund verwendet:

```
<html>
<head>
<style>
body {
background-
image:url('
0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAAZQTFRF%2F6cA%2F%2F%2F%2Fgxp3lwAAAAJ0Uk5T%2FwDltzBKAAA
BF0lEQVR42uzZQQ7CMAxE0e%2F7X5oNCyRocWzPiJbMBZ6qpIljE%2BnwklgKG7kwUjc2IkIaxkY0CPdEsCCasws6ShX
BgmBBmEagpXQQLAgWBAuSY2gaKaWPYEGwIEwg0FRmECwIFoQeQjJlhJWUEFazjFDJCkI5WYRWMgjtfEGYyQnCXD4jTCd
m1zmngFpBFznwVNi5RPSbwbWnpYr%2BBHi%2FtCTfgPLEPL7jBctAKBRptXJ8M%2BprIuZKu%2BUKcg4YK1PLz7kx4bS
qHyPaT4d%2B28OCJJiRBo4FCQsSA0bziT3XubMgYUG6fc5fatmGBQkL0hoJ1IaZMiQsSFiQ8vRscTjlQOI2iHZwtpHuf
%2BJAYiOiJSkj8Z%2FIQ4ABANvXGLd3%2BZMrAAAAAElFTkSuQmCC');
background-repeat:repeat;
}
</style>
</head>
<body>
</body>
</html>
```

Achten Sie bei Verwendung von data:-URLs darauf, dass zusätzliche Leerräume von Bedeutung sind. So muss der Datenstring als einzelne Zeile ohne Unterbrechung eingegeben werden. Andernfalls werden die Zeilenumbrüche als Teil der Daten interpretiert und das Bild kann nicht dekodiert werden.

CSS in AIR

Adobe AIR 1.0 und höher

WebKit unterstützt mehrere erweiterte CSS-Eigenschaften. Viele dieser Erweiterungen verwenden das Präfix: `-webkit`. Beachten Sie, dass einige dieser Erweiterungen zu Versuchszwecken dienen und möglicherweise aus zukünftigen WebKit-Versionen entfernt werden. Weitere Informationen über die Webkit-Unterstützung für CSS und die CSS-Erweiterungen finden Sie unter Safari-CSS-Referenz.

In AIR nicht unterstützte WebKit-Funktionsmerkmale

Adobe AIR 1.0 und höher

AIR bietet keine Unterstützung für die folgenden in WebKit oder Safari 4 verfügbaren Funktionsmerkmale:

- Domänenübergreifende Nachrichtenübermittlung über window.postMessage (AIR verfügt über eigene APIs für die domänenübergreifende Kommunikation)
- CSS-Variablen
- WOFF- (Web Open Font Format) und SVG-Schriftarten
- HTML-Tags „video“ und „audio“
- Mediengeräteabfragen
- Offline-Anwendungscache

- Drucken (AIR stellt eine eigene PrintJob-API bereit)
- Rechtschreib- und Grammatikprüfer
- SVG
- WAI-ARIA
- WebSockets (AIR stellt eigene Socket-APIs bereit)
- Web Worker
- SQL-API von WebKit (AIR verfügt über eine eigene API)
- Geolokation-API von WebKit (AIR verfügt über eine eigene Geolokation-API auf unterstützten Geräten)
- API zum Hochladen mehrerer Dateien von WebKit
- WebKit-Berührungsereignisse (AIR verfügt über eigene Berührungsereignisse)
- WML (Wireless Markup Language)

Die folgenden Listen enthalten spezifische JavaScript-APIs, HTML-Elemente und CSS-Eigenschaften und -Werte, die von AIR nicht unterstützt werden:

Nicht unterstützte JavaScript Window-Objektmitglieder:

- applicationCache()
- console
- openDatabase()
- postMessage()
- document.print()

Nicht unterstützte HTML-Tags:

- audio
- video

Nicht unterstützte HTML-Attribute:

- aria-*
- draggable
- formnovalidate
- list
- novalidate
- onbeforeload
- onhashchange
- onorientationchange
- onpagehide
- onpageshow
- onpopstate
- ontouchstart
- ontouchmove

- ontouchend
- ontouchcancel
- onwebkitbeginfullscreen
- onwebkitendfullscreen
- pattern
- required
- sandbox

Nicht unterstützte JavaScript-Ereignisse:

- beforeload
- hashchange
- orientationchange
- pagehide
- pageshow
- popstate
- touchstart
- touchmove
- touchend
- touchcancel
- webkitbeginfullscreen
- webkitendfullscreen

Nicht unterstützte CSS-Eigenschaften:

- background-clip
- background-origin (verwenden Sie -webkit-background-origin)
- background-repeat-x
- background-repeat-y
- background-size (verwenden Sie -webkit-background-size)
- border-bottom-left-radius
- border-bottom-right-radius
- border-radius
- border-top-left-radius
- border-top-right-radius
- text-rendering
- -webkit-animation-play-state
- -webkit-background-clip
- -webkit-color-correction
- -webkit-font-smoothing

Nicht unterstützte CSS-Werte:

- Eigenschaftswerte für die Darstellung (appearance):
  - media-volume-slider-container
  - media-volume-slider
  - media-volume-sliderthumb
  - outer-spin-button
- border-box (background-clip und background-origin)
- contain (background-size)
- content-box (background-clip und background-origin)
- cover (background-size)
- Eigenschaftswerte für Listen (list):
  - afar
  - amharic
  - amharic-abegede
  - cjk-earthly-branch
  - cjk-heavenly-stem
  - ethiopic
  - ethiopic-abegede
  - ethiopic-abegede-am-et
  - ethiopic-abegede-gez
  - ethiopic-abegede-ti-er
  - ethiopic-abegede-ti-et
  - ethiopic-halehame-aa-er
  - ethiopic-halehame-aa-et
  - ethiopic-halehame-am-et
  - ethiopic-halehame-gez
  - ethiopic-halehame-om-et
  - ethiopic-halehame-sid-et
  - ethiopic-halehame-so-et
  - ethiopic-halehame-ti-er
  - ethiopic-halehame-ti-et
  - ethiopic-halehame-tig
  - hangul
  - hangul-consonant
  - lower-norwegian
  - oromo
  - sidama

  - somali
  - tigre
  - tigrinya-er
  - tigrinya-er-abegede
  - tigrinya-et
  - tigrinya-et-abegede
  - upper-greek
  - upper-norwegian
- -wap-marquee (Anzeigeeigenschaft)

Kapitel 59: Programmieren mit HTML und JavaScript in AIR

Adobe AIR 1.0 und höher

Eine Reihe von Themen zur Programmierung gelten einzig für das Entwickeln von Adobe® AIR®-Anwendungen mit HTML und JavaScript. Die folgenden Informationen sind wichtig, unabhängig davon, ob Sie eine HTML-basierte AIR-Anwendung programmieren oder eine SWF-basierte AIR-Anwendung, die mithilfe der HTMLLoader-Klasse (oder der mx:HTML Flex™-Komponente) HTML und JavaScript ausführt.

HTMLLoader-Klasse

Adobe AIR 1.0 und höher

Die HTMLLoader-Klasse von Adobe AIR definiert das Anzeigeobjekt, das HTML-Inhalt in einer AIR-Anwendung anzeigen kann. SWF-basierte Anwendungen können dem vorhandenen Fenster ein HTMLLoader-Steuerelement hinzufügen oder ein HTML-Fenster erstellen, das automatisch ein HTMLLoader-Objekt mit `HTMLLoader.createRootWindow()` enthält. Der Zugriff auf das HTMLLoader-Objekt kann aus der geladenen HTML-Seite über die `window.htmlLoader`-JavaScript-Eigenschaft erfolgen.

Laden von HTML-Inhalten über eine URL

Adobe AIR 1.0 und höher

Mit dem folgenden Code wird eine URL in ein HTMLLoader-Objekt geladen (fügen Sie den HTMLLoader als ein untergeordnetes Element der Bühne oder eines anderen Anzeigeobjektcontainers hinzu, um den HTML-Inhalt in Ihrer Anwendung anzuzeigen):

```
import flash.html.HTMLLoader;

var html:HTMLLoader = new HTMLLoader;
html.width = 400;
html.height = 600;
var urlReq:URLRequest = new URLRequest("http://www.adobe.com/");
html.load(urlReq);
```

Für die `width`- und `height`-Eigenschaft eines HTMLLoader-Objekts ist standardmäßig „0“ festgelegt. Diese Dimensionen sollten Sie beim Hinzufügen eines HTMLLoader-Objekts zum Stage-Objekt festlegen. Das HTMLLoader-Objekt löst beim Laden der Seite mehrere Ereignisse aus. Anhand dieser Ereignisse können Sie ermitteln, ob eine Interaktion mit der geladenen Seite sicher ist. Diese Ereignisse sind unter „Verarbeiten HTML-bezogener Ereignisse in AIR“ auf Seite 1091 beschrieben.

Hinweis: *In der Flex-Architektur können nur Klassen, die über die UIComponent-Klasse hinausgehen, als untergeordnete Objekte von Flex-Containerkomponenten hinzugefügt werden. Daher können Sie ein HTMLLoader-Objekt nicht direkt als untergeordnetes Objekt einer Flex-Containerkomponente hinzufügen. Mit dem Flex mx:HTML-Steuerelement können Sie jedoch eine benutzerdefinierte Klasse erstellen, die UIComponent erweitert und ein HTMLLoader-Objekt als Unterobjekt von UIComponent enthält, oder Sie können das HTMLLoader-Objekt als Unterobjekt von UIComponent hinzufügen und diese dem Flex-Container hinzufügen.*

Sie können HTML-Text auch mithilfe der TextField-Klasse rendern, diese hat jedoch nur einen begrenzten Funktionsumfang. Die TextField-Klasse von Adobe® Flash® Player unterstützt eine Untergruppe von HTML-Code, doch ist ihr Funktionsumfang aufgrund von Größenbeschränkungen begrenzt. (Die in Adobe AIR enthaltene HTMLLoader-Klasse ist in Flash Player nicht verfügbar.)

Laden von HTML-Inhalten aus einem String

Adobe AIR 1.0 und höher

Die `loadString()`-Methode eines HTMLLoader-Objekts lädt einen String des HTML-Inhalts in das HTMLLoader-Objekt:

```
var html:HTMLLoader = new HTMLLoader();
var htmlStr:String = "<html><body>Hello <b>world</b>.</body></html>";
html.loadString(htmlStr);
```

Standardmäßig wird der über die `loadString()`-Methode geladene Inhalt in einer anwendungsfremden Sandbox mit den folgenden Merkmalen platziert:

- Es kann Inhalt aus dem Netzwerk (nicht jedoch aus dem Dateisystem) geladen werden.
- Mit XMLHttpRequest können keine Daten geladen werden.
- Die `window.location`-Eigenschaft ist mit dem `"about:blank"` belegt.
- Der Inhalt kann nicht auf die `window.runtime`-Eigenschaft zugreifen (wie Inhalt in beliebigen anwendungsfremden Sandboxes dies kann).

In AIR 1.5 enthält die HTMLLoader-Klasse eine `placeLoadStringContentInApplicationSandbox`-Eigenschaft. Wenn diese Eigenschaft für ein HTMLLoader-Objekt auf `true` eingestellt ist, wird Inhalt, der über die `loadString()`-Methode geladen wird, in der Anwendungs-Sandbox platziert. (Der Standardwert lautet `false`.) Damit erhält Inhalt, der über die `loadString()`-Methode geladen wird, Zugriff auf die `window.runtime`-Eigenschaft und auf alle AIR-APIs. Wenn Sie diese Eigenschaft auf `true` einstellen, stellen Sie sicher, dass die Datenquelle für einen String, der in einem Aufruf der `loadString()`-Methode verwendet wird, vertrauenswürdig ist. Code-Anweisungen im HTML-String werden mit vollständigen Anwendungsberechtigungen ausgeführt, wenn diese Eigenschaft den Wert `true` aufweist. Stellen Sie diese Eigenschaft nur dann auf `true`, wenn Sie sicher sein können, dass der String keinen schädigenden Code enthält.

In Anwendungen, die mit dem AIR 1.0- oder AIR 1.1-SDK kompiliert werden, wird über die `loadString()`-Methode geladener Inhalt in der Anwendungs-Sandbox platziert.

Wichtige Sicherheitsregeln beim Verwenden von HTML in AIR-Anwendungen

Adobe AIR 1.0 und höher

Die mit der AIR-Anwendung installierten Dateien haben Zugriff auf die AIR-APIs. Aus Sicherheitsgründen gilt dies nicht für Inhalt aus anderen Quellen. Diese Einschränkung verhindert zum Beispiel, dass Code aus einer Remotedomäne (wie z. B. http://example.com) den Inhalt des Desktopverzeichnisses eines Benutzers (oder sonstige vertrauliche Informationen) liest.

Aufgrund von Sicherheitslücken, die durch Aufrufen der `eval()`-Funktion (und zugehöriger APIs) ausgenutzt werden können, kann mit der Anwendung installierter Inhalt diese Methoden standardmäßig nicht verwenden. Einige Ajax-Architekturen rufen jedoch die `eval()`-Funktion und die zugehörigen APIs auf.

Für die ordnungsgemäße Strukturierung von Inhalt, damit er in einer AIR-Anwendung verwendet werden kann, müssen Sie die Regeln für Sicherheitsbeschränkungen für Inhalt aus unterschiedlichen Quellen berücksichtigen. Inhalt aus unterschiedlichen Quellen wird in separaten, als Sandboxen bezeichneten Sicherheitsklassifizierungen (siehe „Sicherheits-Sandboxen“ auf Seite 1117) abgelegt. Der mit der Anwendung installierte Inhalt wird standardmäßig in einer so genannten *Anwendungs*-Sandbox installiert. Damit hat er Zugriff auf die AIR-APIs. Die Anwendungs-Sandbox ist normalerweise die sicherste aller Sandboxen und ihre Beschränkungen verhindern die Ausführung nicht vertrauenswürdigen Codes.

Die Laufzeitumgebung ermöglicht es Ihnen, mit der Anwendung installierten Inhalt in einer von der Anwendungs-Sandbox abweichenden Sandbox zu laden. Inhalt in anderen Sandboxen befindet sich in einer Sicherheitsumgebung, die der Sicherheitsumgebung eines gängigen Webbrowsers ähnelt. Code in anwendungsfremden Sandboxen kann beispielsweise `eval()` und ähnliche Methoden verwenden (aber nicht auf die AIR-APIs zugreifen). Die Laufzeitumgebung umfasst Möglichkeiten, mit denen Inhalt in anderen Sandboxen sicher kommunizieren kann (z. B. ohne die AIR-APIs anwendungsfremdem Inhalt auszusetzen). Weitere Informationen finden Sie unter „Cross-Scripting von Inhalten in unterschiedlichen Sicherheits-Sandboxen“ auf Seite 1067.

Wenn Sie Code aufrufen, dessen Verwendung in einer Sandbox aus Sicherheitsgründen beschränkt ist, löst die Laufzeitumgebung einen JavaScript-Fehler aus: „Adobe AIR runtime security violation for JavaScript code in the application security sandbox“ (Sicherheitsverstoß gegen Adobe AIR-Laufzeitumgebung für JavaScript-Code in der Sicherheits-Sandbox der Anwendung).

Halten Sie sich an die im nächsten Abschnitt, „Vermeiden von sicherheitsbezogenen JavaScript-Fehlern“ auf Seite 1049, beschriebenen Kodierungspraktiken, um diesen Fehler zu vermeiden.

Weitere Informationen finden Sie unter „HTML-Sicherheit in Adobe AIR“ auf Seite 1158.

Vermeiden von sicherheitsbezogenen JavaScript-Fehlern

Adobe AIR 1.0 und höher

Wenn Sie Code aufrufen, dessen Verwendung in einer Sandbox aus diesen Sicherheitsgründen beschränkt ist, löst die Laufzeitumgebung einen JavaScript-Fehler aus: „Adobe AIR runtime security violation for JavaScript code in the application security sandbox“ (Sicherheitsverstoß gegen Adobe AIR-Laufzeitumgebung für JavaScript-Code in der Sicherheits-Sandbox der Anwendung). Halten Sie sich an die folgenden Kodierungspraktiken, um diesen Fehler zu vermeiden.

Ursachen sicherheitsbezogener JavaScript-Fehler

Adobe AIR 1.0 und höher

Wenn das `load`-Ereignis des Dokuments ausgelöst und beliebige `load`-Ereignisprozeduren beendet wurden, ist in der Anwendungs-Sandbox ausgeführter Code von den meisten Operationen ausgenommen, bei denen Strings bewertet und ausgeführt werden. Bei dem Versuch, die folgenden Typen von JavaScript-Anweisungen zu verwenden, die potenziell unsichere Strings bewerten und ausführen, treten JavaScript-Fehler auf:

- eval()-Funktion
- setTimeout() und setInterval()
- Function-Konstruktor

  Darüber hinaus schlagen die folgenden Typen von JavaScript-Anweisungen fehl, ohne dass ein sicherheitsbezogener JavaScript-Fehler auftritt:

- javascript: URLs
- Ereignisrückrufe, die über die onevent-Attribute in innerHTML- und outerHTML-Anweisungen zugewiesen wurden
- Laden von JavaScript-Dateien aus Verzeichnissen außerhalb des Anwendungsinstallationsverzeichnisses
- document.write() und document.writeln()
- Synchrone XMLHttpRequests vor dem load-Ereignis oder während einer load-Ereignisprozedur
- Dynamisch erstellte Skriptelemente

  Hinweis: *In einigen wenigen Fällen ist die Bewertung von Strings zulässig. Weitere Informationen finden Sie unter „Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen“ auf Seite 1161.*

  Unter http://www.adobe.com/go/airappsandboxframeworks_de finden Sie eine von Adobe erstellte Liste der Ajax-Architekturen, die die Sicherheits-Sandbox der Anwendung unterstützen.

  In den folgenden Abschnitten erfahren Sie, wie Sie Skripts umschreiben können, um diese sicherheitsbezogenen JavaScript-Fehler und ein stillschweigendes Fehlschlagen von Code zu verhindern, der in der Anwendungs-Sandbox ausgeführt wird.

Zuordnen von Anwendungsinhalten zu einer anderen Sandbox

Adobe AIR 1.0 und höher

In den meisten Fällen können Sie eine Anwendung umschreiben oder umstrukturieren, um sicherheitsbezogene JavaScript-Fehler zu vermeiden. Ist ein Umschreiben oder Umstrukturieren nicht möglich, können Sie den Anwendungsinhalt stattdessen mit dem unter „Laden von Anwendungsinhalten in eine anwendungsfremde Sandbox“ auf Seite 1068 beschriebenen Verfahren in eine andere Sandbox laden. Wenn dieser Inhalt außerdem auf AIR-APIs zugreifen muss, können Sie eine Sandbox-Brücke erstellen, wie unter „Einrichten einer Schnittstelle für Sandbox-Brücken“ auf Seite 1068 beschrieben.

eval()-Funktion

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die `eval()`-Funktion kann in der Anwendungs-Sandbox nur vor dem `load`-Ereignis oder während der `load`-Ereignisprozedur verwendet werden. Nach dem Laden der Seite wird durch Aufrufen von `eval()` kein Code ausgeführt. In den folgenden Fällen können Sie den Code jedoch neuschreiben, um die Verwendung von `eval()` zu verhindern.

Zuweisen von Eigenschaften zu Objekten

Adobe AIR 1.0 und höher

Statt einen String zum Erstellen des Eigenschaften-Accessors wie folgt zu analysieren:

```
eval("obj." + propName + " = " + val);
```

Greifen Sie wie folgt auf Eigenschaften mit Klammern zu:

```
obj[propName] = val;
```

Erstellen von Funktionen mit im Kontext verfügbaren Variablen

Adobe AIR 1.0 und höher

Ersetzen Sie Anweisungen wie beispielsweise die folgende:

```
function compile(var1, var2){
    eval("var fn = function(){ this."+var1+"(var2) }");
    return fn;
}
```

Durch:

```
function compile(var1, var2){
    var self = this;
    return function(){ self[var1](var2) };
}
```

Erstellen von Objekten unter Verwendung des Klassennamens als Stringparameter

Adobe AIR 1.0 und höher

Sehen Sie sich eine mit folgendem Code definierte hypothetische JavaScript-Klasse an:

```
var CustomClass =
    {
        Utils:
        {
            Parser: function(){ alert('constructor') }
        },
        Data:
        {

        }
    };
var constructorClassName = "CustomClass.Utils.Parser";
```

Mit `eval()` ließe sich am einfachsten eine Instanz erstellen:

```
var myObj;
eval('myObj=new ' + constructorClassName +'()')
```

Sie können das Aufrufen von `eval()` jedoch vermeiden, indem Sie jede Komponente des Klassennamens analysieren und das neue Objekt durch Verwenden von Klammern erstellen:

```
function getter(str)
{
    var obj = window;
    var names = str.split('.');
    for(var i=0;i<names.length;i++){
        if(typeof obj[names[i]]=='undefined'){
            var undefstring = names[0];
            for(var j=1;j<=i;j++)
                undefstring+="."+names[j];
            throw new Error(undefstring+" is undefined");
        }
        obj = obj[names[i]];
    }
    return obj;
}
```

Verwenden Sie folgenden Code zum Erstellen der Instanz:

```
try{
    var Parser = getter(constructorClassName);
    var a = new Parser();
    }catch(e){
        alert(e);
}
```

setTimeout() und setInterval()

Adobe AIR 1.0 und höher

Ersetzen Sie den an die Prozedurfunktion übergebenen String durch einen Funktionsverweis oder ein Objekt. Ersetzen Sie beispielsweise folgende Anweisung:

```
setTimeout("alert('Timeout')", 100);
```

Durch:

```
setTimeout(function(){alert('Timeout')}, 100);
```

Muss bei der Funktion das `this`-Objekt vom Aufrufer festgelegt werden, ersetzen Sie folgende Anweisung:

```
this.appTimer = setInterval("obj.customFunction();", 100);
```

durch Folgendes:

```
var _self = this;
this.appTimer = setInterval(function(){obj.customFunction.apply(_self);}, 100);
```

Function-Konstruktor

Adobe AIR 1.0 und höher

Aufrufe von `new Function(param, body)` können durch eine Inline-Funktionsdeklaration ersetzt oder nur vor der Verarbeitung des `load`-Ereignisses der Seite verwendet werden.

javascript: URLs

Adobe AIR 1.0 und höher

Der Code in einer Verknüpfung, die das javascript:-URL-Schema verwendet, wird in der Anwendungs-Sandbox ignoriert. Es wird kein sicherheitsbezogener JavaScript-Fehler generiert. Sie können Verknüpfungen z. B. mit folgenden javascript:-URLs ersetzen:

```
<a href="javascript:code()">Click Me</a>
```

Durch:

```
<a href="#" onclick="code()">Click Me</a>
```

Ereignisrückrufe, die über die onevent-Attribute in innerHTML- und outerHTML-Anweisungen zugewiesen wurden

Adobe AIR 1.0 und höher

Wenn Sie dem DOM eines Dokuments mit innerHTML oder outerHTML Elemente hinzufügen, werden alle in der Anweisung zugewiesenen Ereignisrückrufe wie `onclick` oder `onmouseover` ignoriert. Es wird kein Sicherheitsfehler generiert. Stattdessen können Sie den neuen Elementen ein `id`-Attribut zuweisen und die Rückruffunktionen der Ereignisprozedur mit der `addEventListener()`-Methode festlegen.

Enthält ein Dokument z. B. folgendes Zielelement:

```
<div id="container"></div>
```

Ersetzen Sie Anweisungen wie:

```
document.getElementById('container').innerHTML =
    '<a href="#" onclick="code()">Click Me.</a>';
```

Durch:

```
document.getElementById('container').innerHTML = '<a href="#" id="smith">Click Me.</a>';
document.getElementById('smith').addEventListener("click", function() { code(); });
```

Laden von JavaScript-Dateien aus Verzeichnissen außerhalb des Anwendungsinstallationsverzeichnisses

Adobe AIR 1.0 und höher

Das Laden von Skriptdateien von außerhalb der Anwendungs-Sandbox ist nicht zulässig. Es wird kein Sicherheitsfehler generiert. Alle in der Anwendungs-Sandbox ausgeführten Skriptdateien müssen im Anwendungsverzeichnis installiert sein. Wenn in einer Seite externe Skripts verwendet werden sollen, müssen Sie die Seite einer anderen Sandbox zuweisen. Siehe „Laden von Anwendungsinhalten in eine anwendungsfremde Sandbox“ auf Seite 1068.

document.write() und document.writeln()

Adobe AIR 1.0 und höher

Aufrufe von `document.write()` oder `document.writeln()` werden nach Verarbeitung des `load`-Ereignisses ignoriert. Es wird kein Sicherheitsfehler generiert. Alternativ können Sie eine neue Datei laden oder den Dokumentkörper mit DOM-Manipulationsverfahren ersetzen.

Synchrone XMLHttpRequests vor dem load-Ereignis oder während einer load-Ereignisprozedur

Adobe AIR 1.0 und höher

Synchrone XMLHttpRequests, die vor der Durchführung des `load`-Ereignisses einer Seite oder während der Ausführung der `load`-Ereignisprozedur initiiert werden, geben keinen Inhalt zurück. Asynchrone XMLHttpRequests können zwar initiiert werden, geben aber erst nach dem `load`-Ereignis Inhalt zurück. Nach der Verarbeitung des `load`-Ereignisses verhalten sich synchrone XMLHttpRequests wie üblich.

Dynamisch erstellte Skriptelemente

Adobe AIR 1.0 und höher

Dynamisch erstellte Skriptelemente, wie die mit innerHTML oder der `document.createElement()`-Methode erstellten, werden ignoriert.

Zugreifen auf AIR API-Klassen aus JavaScript

Adobe AIR 1.0 und höher

Neben den Standard- und erweiterten Elementen von Webkit kann HTML- und JavaScript-Code auf die von der Laufzeitumgebung bereitgestellten Hostklassen zugreifen. Mit diesen Klassen können Sie auf die von AIR bereitgestellten, erweiterten Funktionen zugreifen, darunter:

- Zugriff auf das Dateisystem
- Verwendung lokaler SQL-Datenbanken
- Steuerung von Anwendung- und Fenstermenüs

- Zugriff auf Sockets für Netzwerke
- Verwendung benutzerdefinierter Klassen und Objekte
- Soundfunktionen

Die Datei-API von AIR umfasst beispielsweise eine File-Klasse, die im flash.filesystem-Paket enthalten ist. In JavaScript können Sie ein File-Objekt wie folgt erstellen:

```
var myFile = new window.runtime.flash.filesystem.File();
```

Das `runtime`-Objekt ist ein spezielles JavaScript-Objekt, das in AIR in der Anwendungs-Sandbox ausgeführtem HTML-Inhalt zur Verfügung steht. Mit diesem können Sie auf Laufzeitklassen aus JavaScript zugreifen. Die `flash`-Eigenschaft des `runtime`-Objekts bietet Zugriff auf das flash-Paket. Die `flash.filesystem`-Eigenschaft des `runtime`-Objekts bietet wiederum Zugriff auf das flash.filesystem-Paket (das die File-Klasse enthält). Pakete dienen zum Organisieren von in ActionScript verwendeten Klassen.

Hinweis: *Die `runtime`-Eigenschaft wird den Fensterobjekten von Seiten, die in einen Frame oder Inlineframe geladen wurden, nicht automatisch hinzugefügt. Solange das untergeordnete Dokument sich in der Anwendungs-Sandbox befindet, kann das untergeordnete Objekt auf die `runtime`-Eigenschaft des übergeordneten Objekts zugreifen.*

Da Entwickler aufgrund der Paketstruktur der Laufzeitklassen für den Zugriff auf jede Klasse lange Strings in JavaScript-Code eingeben müssten (z. B. `window.runtime.flash.desktop.NativeApplication`), enthält das AIR-SDK die Datei„ AIRAliases.js“, mit der Sie wesentlich einfacher auf Laufzeitklassen zugreifen können (z. B. durch Eingeben von `air.NativeApplication`).

Die AIR API-Klassen werden an verschiedenen Stellen in diesem Handbuch behandelt. Andere Klassen aus der Flash Player-API, die für HTML-Entwickler interessant sein könnten, werden in der *Adobe AIR API Reference for HTML Developers* beschrieben. ActionScript ist die in SWF-Inhalt (Flash Player) verwendete Sprache. Allerdings ähneln sich die JavaScript- und ActionScript-Syntax. (Beide basieren auf Versionen der ECMAScript-Sprache.) Alle integrierten Klassen sind sowohl in JavaScript (in HTML-Inhalt) als auch in ActionScript (in SWF-Inhalt) verfügbar.

Hinweis: *JavaScript-Code kann die in ActionScript verfügbare Dictionary-, XML- und XMLList-Klasse nicht verwenden.*

Verwenden der AIRAliases.js-Datei

Adobe AIR 1.0 und höher

Die Laufzeitklassen sind wie im Folgenden in einer Paketstruktur organisiert:

- `window.runtime.flash.desktop.NativeApplication`
- `window.runtime.flash.desktop.ClipboardManager`
- `window.runtime.flash.filesystem.FileStream`
- `window.runtime.flash.data.SQLDatabase`

Das AIR-SDK enthält die Datei „AIRAliases.js“, die „Alias“-Definitionen bereitstellt, mit denen Sie auf Laufzeitklassen zugreifen können, ohne lange Strings einzugeben. Sie können beispielsweise durch folgende Eingaben auf die oben aufgeführten Klassen zugreifen:

- `air.NativeApplication`
- `air.Clipboard`
- `air.FileStream`

- `air.SQLDatabase`

  Diese Liste enthält lediglich einen kleinen Teil der in der Datei „AIRAliases.js“ verfügbaren Klassen. Eine vollständige Liste von Klassen und Funktionen auf Paketebene finden Sie in der *Adobe AIR API Reference for HTML Developers*.

  Neben den gängigen Laufzeitklassen enthält die Datei „AIRAliases.js“ außerdem Aliase für gängige Funktionen auf Paketebene: `window.runtime.trace()`, `window.runtime.flash.net.navigateToURL()` und `window.runtime.flash.net.sendToURL()`, deren Aliase `air.trace()`, `air.navigateToURL()` und `air.sendToURL()` sind.

  Um die Datei „AIRAliases.js“ zu verwenden, fügen Sie den folgenden `script`-Verweis in der HTML-Seite hinzu:

```
<script src="AIRAliases.js"></script>
```

  Passen Sie den Pfad im `src`-Verweis nach Bedarf an.

  Wichtig: *Sofern nicht ausdrücklich angegeben, wird beim JavaScript-Beispielcode in dieser Dokumentation davon ausgegangen, dass Sie die Datei „AIRAliases.js“ in der HTML-Seite aufgenommen haben.*

URLs in AIR

Adobe AIR 1.0 und höher

In HTML-Inhalt, der in AIR ausgeführt wird, können Sie ein beliebiges der folgenden URL-Schemas verwenden, indem Sie `src`-Attribute für den `img`-, `frame`-, `iframe`- und `script`-Tag im `href`-Attribut eines `link`-Tags oder an einer anderen Stelle definieren, an der Sie eine URL bereitstellen können.

URL-Schema	Beschreibung	Beispiel für
file	Ein Pfad relativ zum Stamm des Dateisystems.	`file:///c:/AIR Test/test.txt`
app	Ein Pfad relativ zum Stammverzeichnis der installierten Anwendung.	`app:/images`
app-storage	Ein Pfad relativ zum Anwendungsspeicherverzeichnis. Für jede installierte Anwendung definiert AIR ein eindeutiges Anwendungsspeicherverzeichnis, ein nützlicher Ort zum Speichern anwendungsspezifischer Daten.	`app-storage:/settings/prefs.xml`
http	Eine Standard-HTTP-Anforderung.	`http://www.adobe.com`
https	Eine Standard-HTTPS-Anforderung.	`https://secure.example.com`

Weitere Informationen zum Verwenden von URL-Schemas in AIR finden Sie unter „URI-Schemas“ auf Seite 867.

Viele AIR-APIs, darunter die File-, Loader-, URLStream- und Sound-Klasse, verwenden anstelle eines Strings, der die URL enthält, ein URLRequest-Objekt. Das URLRequest-Objekt selbst wird mit einem String initialisiert, der ein beliebiges, identisches URL-Schema verwenden kann. Die folgende Anweisung erstellt z. B. ein URLRequest-Objekt, das zum Anfordern der Adobe-Homepage verwendet werden kann:

```
var urlReq = new air.URLRequest("http://www.adobe.com/");
```

Informationen zu URLRequest-Objekten finden Sie unter „HTTP-Kommunikation“ auf Seite 865.

Bereitstellen von ActionScript-Objekten für JavaScript

Adobe AIR 1.0 und höher

JavaScript in der von einem HTMLLoader-Objekt geladenen HTML-Seite kann die im ActionScript-Ausführungskontext definierten Klassen, Objekte und Funktionen mit der `window.runtime`-, `window.htmlLoader`- und `window.nativeWindow`-Eigenschaft der HTML-Seite aufrufen. Sie können ActionScript-Objekte und -Funktionen außerdem durch Erstellen von Verweisen auf diese im JavaScript-Ausführungskontext für JavaScript-Code bereitstellen.

Ein einfaches Beispiel zum Zugreifen auf JavaScript-Objekte mit ActionScript

Adobe AIR 1.0 und höher

Das folgende Beispiel veranschaulicht, wie Eigenschaften, die auf ActionScript-Objekte verweisen, dem globalen Fensterobjekt einer HTML-Seite hinzugefügt werden:

```
var html:HTMLLoader = new HTMLLoader();
var foo:String = "Hello from container SWF."
function helloFromJS(message:String):void {
    trace("JavaScript says:", message);
}
var urlReq:URLRequest = new URLRequest("test.html");
html.addEventListener(Event.COMPLETE, loaded);
html.load(urlReq);

function loaded(e:Event):void{
    html.window.foo = foo;
    html.window.helloFromJS = helloFromJS;
}
```

Der im vorhergehenden Beispiel in das HTMLLoader-Objekt geladene HTML-Inhalt (in der Datei „test.html“) kann auf die in der übergeordneten SWF-Datei definierte `foo`-Eigenschaft und `helloFromJS()`-Methode zugreifen:

```
<html>
    <script>
        function alertFoo() {
            alert(foo);
        }
    </script>
    <body>
        <button onClick="alertFoo()">
            What is foo?
        </button>
        <p><button onClick="helloFromJS('Hi.')">
            Call helloFromJS() function.
        </button></p>
    </body>
</html>
```

Beim Zugreifen auf den JavaScript-Kontext eines ladenden Dokuments können Sie mithilfe des `htmlDOMInitialize`-Ereignisses Objekte so früh in der Seitenerstellungsfolge erstellen, dass in der Seite definierte Skripts auf sie zugreifen können. Wenn Sie das `complete`-Ereignis abwarten, können nur Skripts in der Seite, die nach dem `load`-Ereignis der Seite ausgeführt werden, auf die hinzugefügten Objekte zugreifen.

Bereitstellen von Klassendefinitionen für JavaScript

Adobe AIR 1.0 und höher

Um die ActionScript-Klassen der Anwendung in JavaScript bereitzustellen, können Sie den geladenen HTML-Inhalt der Anwendungsdomäne zuweisen, die die Klassendefinitionen enthält. Die Anwendungsdomäne des JavaScript-Ausführungskontexts kann mit der `runtimeApplicationDomain`-Eigenschaft des HTMLLoader-Objekts festgelegt werden. Um die primäre Anwendungsdomäne als Anwendungsdomäne festzulegen, legen Sie für `runtimeApplicationDomainApplicationDomain.currentDomain` fest, wie im folgenden Code veranschaulicht:

```
html.runtimeApplicationDomain = ApplicationDomain.currentDomain;
```

Wenn die `runtimeApplicationDomain`-Eigenschaft festgelegt wurde, verwendet der JavaScript-Kontext die gleichen Klassendefinitionen wie die zugewiesene Domäne. Um eine Instanz einer benutzerdefinierten Klasse in JavaScript zu erstellen, verweisen Sie über die `window.runtime`-Eigenschaft auf die Klassendefinition und verwenden Sie den `new`-Operator:

```
var customClassObject = new window.runtime.CustomClass();
```

Der HTML-Inhalt muss aus einer kompatiblen Sicherheitsdomäne stammen. Stammt der HTML-Inhalt aus einer anderen Sicherheitsdomäne als der zugewiesenen Anwendungsdomäne, verwendet die Seite stattdessen die Standardanwendungsdomäne. Wenn Sie zum Beispiel eine Remoteseite aus dem Internet laden, könnten Sie ApplicationDomain.currentDomain nicht als Anwendungsdomäne der Seite zuweisen.

Entfernen von Ereignisprozeduren

Adobe AIR 1.0 und höher

Wenn Sie Objekten außerhalb der aktuellen Seite, einschließlich Laufzeitobjekte in geladenem SWF-Inhalt und sogar auf anderen Seiten ausgeführte JavaScript-Objekte, JavaScript-Ereignisprozeduren hinzufügen, sollten Sie diese Ereignis-Listener beim Entladen der Datei immer entfernen. Andernfalls sendet der Ereignis-Listener das Ereignis an eine nicht mehr vorhandene Prozedurfunktion. In diesem Fall wird die folgende Fehlermeldung angezeigt: „The application attempted to reference a JavaScript object in an HTML page that is no longer loaded.“ (Die Anwendung hat versucht, auf ein JavaScript-Objekt in einer HTML-Seite zu verweisen, die nicht mehr geladen ist.) Durch Entfernen nicht benötigter Ereignisprozeduren kann AIR den zugehörigen Speicher freigeben. Weitere Informationen finden Sie unter „Entfernen von Ereignis-Listenern auf navigierenden HTML-Seiten“ auf Seite 1096.

Zugreifen auf HTML-DOM- und JavaScript-Objekte mit ActionScript

Adobe AIR 1.0 und höher

Wenn das HTMLLoader-Objekt das `complete`-Ereignis auslöst, können Sie auf alle Objekte im HTML-DOM (Dokumentobjektmodell) der Seite zugreifen. Zu den zugreifbaren Objekten gehören Anzeigeelemente (z. B. das `div`- und `p`-Objekt in der Seite) sowie JavaScript-Variablen und -Funktionen. Das `complete`-Ereignis entspricht dem `load`-Ereignis der JavaScript-Seite. Vor dem Auslösen des `complete`-Ereignisses wurden DOM-Elemente, Variablen und Funktionen möglicherweise nicht analysiert oder erstellt. Warten Sie nach Möglichkeit das `complete`-Ereignis ab, bevor Sie auf das HTML-DOM zugreifen.

Sehen Sie sich beispielsweise folgende HTML-Seite an:

```
<html>
    <script>
        foo = 333;
        function test() {
            return "OK.";
        }
    </script>
    <body>
        <p id="p1">Hi.</p>
    </body>
</html>
```

Diese einfache HTML-Seite definiert eine JavaScript-Variable namens *foo* und eine JavaScript-Funktion namens *test()*. Bei beiden handelt es sich um Eigenschaften des globalen `window`-Objekts der Seite. Darüber hinaus umfasst das `window.document`-Objekt ein benanntes P-Element (mit der ID *p1*), auf das Sie mit der `getElementById()`-Methode zugreifen können. Nachdem die Seite geladen wurde (d. h., wenn das HTMLLoader-Objekt das `complete`-Ereignis auslöst), können Sie mit ActionScript auf jedes dieser Objekte zugreifen. Siehe dazu den folgenden ActionScript-Code:

```
var html:HTMLLoader = new HTMLLoader();
html.width = 300;
html.height = 300;
html.addEventListener(Event.COMPLETE, completeHandler);
var xhtml:XML =
    <html>
        <script>
            foo = 333;
            function test() {
                return "OK.";
            }
        </script>
        <body>
            <p id="p1">Hi.</p>
        </body>
    </html>;
html.loadString(xhtml.toString());

function completeHandler(e:Event):void {
    trace(html.window.foo); // 333
    trace(html.window.document.getElementById("p1").innerHTML); // Hi.
    trace(html.window.test()); // OK.
}
```

Verwenden Sie zum Zugreifen auf den Inhalt eines HTML-Elements die `innerHTML`-Eigenschaft. Der vorhergehende Code verwendet z. B. `html.window.document.getElementById("p1").innerHTML` zum Abrufen des Inhalts eines HTML-Elements namens *p1*.

Sie können Eigenschaften der HTML-Seite auch mit ActionScript festlegen. Das folgende Beispiel legt den Inhalt des `p1`-Elements und den Wert der `foo`-JavaScript-Variablen auf der Seite mit einem Verweis fest, der das HTMLLoader-Objekt enthält:

```
html.window.document.getElementById("p1").innerHTML = "Goodbye";
html.window.foo = 66;
```

Einbetten von SWF-Inhalten in HTML

Adobe AIR 1.0 und höher

Sie können beim Einbetten von SWF-Inhalt in HTML-Inhalt in einer AIR-Anwendung genauso verfahren wie in einem Browser. Betten Sie den SWF-Inhalt mit einem `object`-Tag, einem `embed`-Tag oder beiden in.

Hinweis: *Bei der Webentwicklung ist es üblich, sowohl einen `object`-Tag als auch einen `embed`-Tag zum Anzeigen von SWF-Inhalt in einer HTML-Seite zu verwenden. Diese Praxis bringt in AIR keinerlei Vorteile. Sie können das dem W3C-Standard entsprechende `object`-Tag allein in Inhalt verwenden, der in AIR angezeigt werden soll. Dennoch können Sie bei Bedarf das `object`- und den `embed`-Tag weiterhin für HTML -Inhalt verwenden, der zusätzlich in einem Browser angezeigt werden soll.*

Wenn Sie die Transparenz für das NativeWindow-Objekt, in dem der HTML- und SWF-Inhalt angezeigt wird, aktiviert haben, zeigt AIR den SWF-Inhalt nicht an, wenn der Fenstermodus (`wmode`) zum Einbetten des Inhalts auf den Wert `window` eingestellt ist. Zum Anzeigen von SWF-Inhalt in einer HTML-Seite eines transparenten Fensters muss der `wmode`-Parameter auf `opaque` oder `transparent` eingestellt werden. Da `window` der Standardwert für `wmode` ist, wird der Inhalt möglicherweise nicht angezeigt, wenn Sie keinen Wert angeben.

Das folgende Beispiel veranschaulicht die Verwendung des HTML-`object`-Tags zum Anzeigen einer SWF-Datei in HTML-Inhalt. Der `wmode`-Parameter ist auf `opaque` eingestellt, sodass der Inhalt auch dann angezeigt wird, wenn das zugrunde liegende NativeWindow-Objekt transparent ist. Die SWF-Datei wird aus dem Anwendungsverzeichnis geladen, Sie können allerdings ein beliebiges, von AIR unterstütztes URL-Schema verwenden. (Der Ort, von dem die SWF-Datei geladen wird, bestimmt die Sicherheits-Sandbox, in der AIR den Inhalt ablegt.)

```
<object type="application/x-shockwave-flash" width="100%" height="100%">
    <param name="movie" value="app:/SWFFile.swf"></param>
    <param name="wmode" value="opaque"></param>
</object>
```

Mit einem Skript können Sie Inhalt außerdem dynamisch laden. Das folgende Beispiel erstellt einen `object`-Knoten zum Anzeigen der im `urlString`-Parameter angegebenen SWF-Datei. Der Knoten wird als untergeordnetes Element des Seitenelements hinzugefügt, dessen ID vom `elementID`-Parameter angegeben wird:

```
<script>
function showSWF(urlString, elementID){
    var displayContainer = document.getElementById(elementID);
    var flash = createSWFObject(urlString, 'opaque', 650, 650);
    displayContainer.appendChild(flash);
}
function createSWFObject(urlString, wmodeString, width, height){
    var SWFObject = document.createElement("object");
    SWFObject.setAttribute("type","application/x-shockwave-flash");
    SWFObject.setAttribute("width","100%");
    SWFObject.setAttribute("height","100%");
    var movieParam = document.createElement("param");
    movieParam.setAttribute("name","movie");
    movieParam.setAttribute("value",urlString);
    SWFObject.appendChild(movieParam);
    var wmodeParam = document.createElement("param");
    wmodeParam.setAttribute("name","wmode");
    wmodeParam.setAttribute("value",wmodeString);
    SWFObject.appendChild(wmodeParam);
    return SWFObject;
}
</script>
```

SWF-Inhalt wird nicht angezeigt, wenn das HTMLLoader-Objekt skaliert oder gedreht wurde oder wenn die `alpha`-Eigenschaft auf einen anderen Wert als 1.0 eingestellt ist. Vor AIR 1.5.2 wurde SWF-Inhalt grundsätzlich nicht in einem transparenten Fenster angezeigt, unabhängig davon, auf welchen Wert `wmode` eingestellt war.

Hinweis: *Wenn ein eingebettetes SWF-Objekt versucht, ein externes Element wie eine Videodatei zu laden, wird der SWF-Inhalt möglicherweise nicht richtig dargestellt, wenn in der HTML-Datei kein absoluter Pfad zur Videodatei angegeben wurde. Ein eingebettetes SWF-Objekt kann jedoch eine externe Bilddatei über einen relativen Pfad laden.*

Das folgende Beispiel zeigt, wie externe Elemente über ein SWF-Objekt geladen werden können, das in HTML-Inhalt eingebettet ist:

```
var imageLoader;

function showSWF(urlString, elementID){
    var displayContainer = document.getElementById(elementID);
    imageLoader = createSWFObject(urlString,650,650);
    displayContainer.appendChild(imageLoader);
}

function createSWFObject(urlString, width, height){

    var SWFObject = document.createElement("object");
    SWFObject.setAttribute("type","application/x-shockwave-flash");
    SWFObject.setAttribute("width","100%");
    SWFObject.setAttribute("height","100%");

    var movieParam = document.createElement("param");
    movieParam.setAttribute("name","movie");
    movieParam.setAttribute("value",urlString);
    SWFObject.appendChild(movieParam);

    var flashVars = document.createElement("param");
    flashVars.setAttribute("name","FlashVars");

    //Load the asset inside the SWF content.
    flashVars.setAttribute("value","imgPath=air.jpg");
    SWFObject.appendChild(flashVars);

    return SWFObject;
}
function loadImage()
{
  showSWF("ImageLoader.swf", "imageSpot");

}
```

Im folgenden ActionScript-Beispiel wird der von der HTML-Datei übergebene Pfad der Bilddatei gelesen und das Bild wird auf der Bühne geladen:

```
package
{
      import flash.display.Sprite;
      import flash.display.LoaderInfo;
      import flash.display.StageScaleMode;
      import flash.display.StageAlign;
      import flash.display.Loader;
      import flash.net.URLRequest;

      public class ImageLoader extends Sprite
      {
            public function ImageLoader()
            {

                  var flashvars = LoaderInfo(this.loaderInfo).parameters;

                  if(flashvars.imgPath){
                        var imageLoader = new Loader();
                        var image = new URLRequest(flashvars.imgPath);
                        imageLoader.load(image);
                        addChild(imageLoader);
                        imageLoader.x = 0;
                        imageLoader.y = 0;
                        stage.scaleMode=StageScaleMode.NO_SCALE;
                        stage.align=StageAlign.TOP_LEFT;
                  }
            }
      }
}
```

Verwenden von ActionScript-Bibliotheken in HTML-Seiten

Adobe AIR 1.0 und höher

AIR erweitert das HTML-Skriptelement so, dass die Seite ActionScript-Klassen in eine kompilierte SWF-Datei importieren kann. Um die im Unterverzeichnis `lib` des Stammanwendungsordners enthaltene Bibliothek *myClasses.swf* zu importieren, müssen Sie das folgende Skript-Tag in einer HTML-Datei hinzufügen:

```
<script src="lib/myClasses.swf" type="application/x-shockwave-flash"></script>
```

Wichtig: *Damit die Bibliothek ordnungsgemäß geladen wird, muss das Typattribut* `type="application/x-shockwave-flash"` *lauten.*

Wenn der SWF-Inhalt als Flash Player 10- oder AIR 1.5-SWF kompiliert wird, müssen Sie den XML-Namespace der Anwendungsdeskriptordatei auf den AIR 1.5-Namespace einstellen.

Das Verzeichnis `lib` und die Datei `myClasses.swf` müssen ebenfalls beim Verpacken der AIR-Datei vorhanden sein.

Greifen Sie über die `runtime`-Eigenschaft des JavaScript-Window-Objekts auf die importierten Klassen zu:

```
var libraryObject = new window.runtime.LibraryClass();
```

Sind die Klassen in der SWF-Datei in Paketen organisiert, müssen Sie auch den Paketnamen aufnehmen. Ist beispielsweise die LibraryClass-Definition in einem Paket namens *utilities* enthalten, müssen Sie mit der folgenden Anweisung eine Instanz der Klasse erstellen:

```
var libraryObject = new window.runtime.utilities.LibraryClass();
```

Hinweis: *Verwenden Sie den* `acompc`*-Compiler, um eine ActionScript-SWF-Bibliothek zu kompilieren, die im Rahmen einer HTML-Seite in AIR verwendet werden soll. Das acompc-Dienstprogramm ist im Lieferumfang des Flex-SDK enthalten und wird in der Flex-SDK-Dokumentation beschrieben.*

Zugreifen auf die HTML-DOM- und JavaScript-Objekte aus einer importierten ActionScript-Datei

Adobe AIR 1.0 und höher

Um auf Objekte in einer HTML-Seite mit ActionScript in einer SWF-Datei zuzugreifen, die mit dem `<script>`-Tag importiert wurde, übergeben Sie einen Verweis auf ein JavaScript-Objekt (wie z. B. `window` oder `document`) an eine im ActionScript-Code definierte Funktion. Verwenden Sie den Verweis in der Funktion zum Zugreifen auf das JavaScript-Objekt (oder andere Objekte, auf die über den übergebenen Verweis zugegriffen werden kann).

Sehen Sie sich beispielsweise folgende HTML-Seite an:

```
<html>
    <script src="ASLibrary.swf" type="application/x-shockwave-flash"></script>
    <script>
        num = 254;
        function getStatus() {
            return "OK.";
        }
        function runASFunction(window){
            var obj = new runtime.ASClass();
            obj.accessDOM(window);
        }
    </script>
    <body onload="runASFunction">
        <p id="p1">Body text.</p>
    </body>
</html>
```

Diese einfache HTML-Seite hat eine JavaScript-Variable namens *num* und eine JavaScript-Funktion namens *getStatus()*. Bei beiden handelt es sich um Eigenschaften des `window`-Objekts der Seite. Das `window.document`-Objekt enthält außerdem ein benanntes P-Element (mit der ID *p1*).

Die Seite lädt eine ActionScript-Datei, „ASLibrary.swf“, die die ASClass-Klasse enthält. ASClass definiert eine Funktion namens `accessDOM()`, die einfach die Werte dieser JavaScript-Objekte verfolgt. Die `accessDOM()`-Methode akzeptiert das JavaScript Window-Objekt als Argument. Mit diesem Window-Verweis kann es auf andere Objekte in der Seite zugreifen, darunter Variablen, Funktionen und DOM-Elemente. Siehe dazu die folgende Definition:

```
public class ASClass{
    public function accessDOM(window:*):void {
        trace(window.num); // 254
        trace(window.document.getElementById("p1").innerHTML); // Body text..
        trace(window.getStatus()); // OK.
    }
}
```

Sie können Eigenschaften der HTML-Seite aus einer importierten ActionScript-Klasse abrufen und festlegen. Die folgende Funktion legt beispielsweise die Inhalte des `p1`-Elements auf der Seite und den Wert der `foo`-JavaScript-Variable auf der Seite fest:

```
public function modifyDOM(window:*):void {
    window.document.getElementById("p1").innerHTML = "Bye";
    window.foo = 66;
```

Konvertieren von Date- und RegExp-Objekten

Adobe AIR 1.0 und höher

Die Sprachen JavaScript und ActionScript definieren beide Date- und RegExp-Klassen, doch werden Objekte dieser Typen normalerweise nicht automatisch zwischen den beiden Ausführungskontexten konvertiert. Sie müssen die Date- und RegExp-Objekte in den entsprechenden Typ konvertieren, bevor Sie sie zum Festlegen von Eigenschaften oder Funktionsparametern in dem anderen Ausführungskontext verwenden.

Der folgende ActionScript-Code konvertiert beispielsweise ein JavaScript-Date-Objekt namens `jsDate` in ein ActionScript-Date-Objekt:

```
var asDate:Date = new Date(jsDate.getMilliseconds());
```

Der folgende ActionScript-Code konvertiert ein JavaScript-RegExp-Objekt namens `jsRegExp` in ein ActionScript-RegExp-Objekt:

```
var flags:String = "";
if (jsRegExp.dotAll) flags += "s";
if (jsRegExp.extended) flags += "x";
if (jsRegExp.global) flags += "g";
if (jsRegExp.ignoreCase) flags += "i";
if (jsRegExp.multiline) flags += "m";
var asRegExp:RegExp = new RegExp(jsRegExp.source, flags);
```

Manipulieren eines HTML-Stylesheets mit ActionScript

Adobe AIR 1.0 und höher

Nachdem das HTMLLoader-Objekt das `complete`-Ereignis ausgelöst hat, können Sie CSS-Stile in einer Seite untersuchen und manipulieren.

Sehen Sie sich beispielsweise das folgende einfache HTML-Dokument an:

```
<html>
<style>
    .style1A { font-family:Arial; font-size:12px }
    .style1B { font-family:Arial; font-size:24px }
</style>
<style>
    .style2 { font-family:Arial; font-size:12px }
</style>
<body>
    <p class="style1A">
        Style 1A
    </p>
    <p class="style1B">
        Style 1B
    </p>
    <p class="style2">
        Style 2
    </p>
</body>
</html>
```

Nachdem ein HTMLLoader-Objekt diesen Inhalt lädt, können Sie die CSS-Stile in der Seite wie hier gezeigt über das `cssRules`-Array des `window.document.styleSheets`-Arrays manipulieren:

```
var html:HTMLLoader = new HTMLLoader( );
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.addEventListener(Event.COMPLETE, completeHandler);
function completeHandler(event:Event):void {
    var styleSheet0:Object = html.window.document.styleSheets[0];
    styleSheet0.cssRules[0].style.fontSize = "32px";
    styleSheet0.cssRules[1].style.color = "#FF0000";
    var styleSheet1:Object = html.window.document.styleSheets[1];
    styleSheet1.cssRules[0].style.color = "blue";
    styleSheet1.cssRules[0].style.font-family = "Monaco";
}
```

Dieser Code passt die CSS-Stile so an, dass das resultierende HTML-Dokument folgendermaßen aussieht:

Style 1A

Style 1B

Style 2

Denken Sie daran, dass Code einer Seite Stile hinzufügen kann, nachdem das HTMLLoader-Objekt das `complete`-Ereignis ausgelöst hat.

Cross-Scripting von Inhalten in unterschiedlichen Sicherheits-Sandboxen

Adobe AIR 1.0 und höher

Das Laufzeitsicherheitsmodell isoliert Code unterschiedlicher Herkunft. Durch Cross-Scripting von Inhalt in unterschiedlichen Sicherheits-Sandboxen können Sie Inhalt in einer Sicherheits-Sandbox den Zugriff auf ausgewählte Eigenschaften und Methoden in einer anderen Sandbox gestatten.

AIR-Sicherheits-Sandboxen und JavaScript-Code

Adobe AIR 1.0 und höher

AIR setzt Herkunftsrichtlinien durch, die verhindern, dass Code aus einer Domäne mit Inhalt in einer anderen Domäne interagiert. Alle Dateien werden entsprechend ihrer Herkunft in einer Sandbox abgelegt. Normalerweise kann Inhalt in der Anwendungs-Sandbox nicht gegen das Herkunftsprinzip verstoßen und Cross-Script-Inhalt nicht von außerhalb des Installationsverzeichnisses der Anwendung geladen werden. AIR bietet allerdings einige Verfahren zum Cross-Scripting anwendungsfremden Inhalts.

Eines dieser Verfahren ordnet Anwendungsinhalt mit Frames oder Inlineframes einer anderen Sicherheits-Sandbox zu. Einige aus dem Sandbox-Bereich der Anwendung geladenen Seiten verhalten sich so, als seien sie aus der Remotedomäne geladen worden. Beim Zuordnen von Anwendungsinhalt zur Domäne *example.com* könnte dieser Inhalt beispielsweise zum Cross-Scripting von Seiten führen, die aus example.com geladen wurden.

Da dieses Verfahren den Anwendungsinhalt in einer anderen Sandbox ablegt, unterliegt Code mit diesem Inhalt nicht mehr den Beschränkungen für die Ausführung von Code in bewerteten Strings. Mit diesem Zuordnungsverfahren für Sandboxen können sie die Beschränkungen selbst dann aufheben, wenn das Cross-Scripting von Remoteinhalt nicht nötig ist. Das Zuordnen von Inhalt auf diese Weise kann insbesondere bei der Arbeit mit einer der zahlreichen JavaScript-Architekturen oder mit bestehendem Code von Nutzen sein, für den die Bewertung von Strings erforderlich ist. Sie sollten jedoch das zusätzliche Risiko bedenken, dass beim Ausführen von Inhalt außerhalb der Anwendungs-Sandbox nicht vertrauenswürdiger Inhalt eingeführt und ausgeführt werden könnte, und sich dagegen absichern.

Gleichzeitig verliert einer anderen Sandbox zugeordneter Anwendungsinhalt seinen Zugriff auf die AIR-APIs, sodass mit dem Zuordnungsverfahren für Sandboxen AIR-Funktionen nicht für außerhalb der Anwendungs-Sandbox ausgeführten Code offen gelegt werden können.

Ein weiteres Cross-Scripting-Verfahren ermöglicht Ihnen die Erstellung einer als *Sandbox-Brücke* bezeichneten Schnittstelle zwischen Inhalt in einer anwendungsfremden Sandbox und dem übergeordneten Dokument in der Anwendungs-Sandbox. Mit dieser Brücke kann der untergeordnete Inhalt auf vom übergeordneten Inhalt definierte Eigenschaften und Methoden und/oder der übergeordnete Inhalt auf vom untergeordneten Inhalt definierte Eigenschaften und Methoden zugreifen.

Und schließlich können Sie domänenübergreifende XMLHttpRequests über die Anwendungs-Sandbox und, wahlweise, über andere Sandboxen durchführen.

Weitere Informationen finden Sie unter „HTML-Frame- und iFrame-Elemente“ auf Seite 1038, „HTML-Sicherheit in Adobe AIR“ auf Seite 1158 und „Das XMLHttpRequest-Objekt“ auf Seite 1032.

Laden von Anwendungsinhalten in eine anwendungsfremde Sandbox

Adobe AIR 1.0 und höher

Um Anwendungsinhalt das sichere Cross-Scripting von Inhalt zu ermöglichen, der von außerhalb des Installationsverzeichnisses der Anwendung geladen wurde, können Sie Anwendungsinhalt mit dem `frame`- oder `iframe`-Element in die gleiche Sicherheits-Sandbox laden wie den externen Inhalt. Wenn das Cross-Scripting von Remoteinhalt nicht erforderlich ist, Sie eine Seite Ihrer Anwendung aber dennoch außerhalb der Anwendungs-Sandbox laden möchten, können Sie mit dem gleichen Verfahren `http://localhost/` oder einen anderen unverfänglichen Wert als Herkunftsdomäne angeben.

AIR fügt dem frame-Element, mit dem Sie festlegen können, ob eine in das frame-Elemente geladene Anwendungsdatei einer anwendungsfremden Sandbox zugeordnet werden soll, die neuen Attribute `sandboxRoot` und `documentRoot` hinzu. In einen Pfad unterhalb der `sandboxRoot`-URL aufgelöste Dateien werden stattdessen aus dem Verzeichnis `documentRoot` geladen. Aus Sicherheitsgründen wird auf diese Weise geladener Anwendungsinhalt so behandelt, als sei er tatsächlich aus der `sandboxRoot`-URL geladen worden.

Die `sandboxRoot`-Eigenschaft gibt die URL an, mit der die Sandbox und Domäne bestimmt werden, in der der Frame-Inhalt abgelegt werden soll. Dazu muss das URL-Schema `file:`, `http:` oder `https:` verwendet werden. Wenn Sie eine relative URL angeben, verbleibt der Inhalt in der Anwendungs-Sandbox.

Die `documentRoot`-Eigenschaft gibt das Verzeichnis an, aus dem der Frame-Inhalt geladen wird. Dazu muss das URL-Schema `file:`, `app:` oder `app:` verwendet werden.

Das folgende Beispiel ordnet im Unterverzeichnis `sandbox` der Anwendung installierten Inhalt für die Ausführung in der Remote-Sandbox und der Domäne `www.example.com` zu:

```
 <iframe
    src="http://www.example.com/local/ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

Die Seite `ui.html` könnte mit folgendem Skript-Tag eine JavaScript-Datei aus dem lokalen Ordner `sandbox` laden:

```
<script src="http://www.example.com/local/ui.js"></script>
```

Mit einem Script-Tag wie dem Folgenden könnte sie ferner Inhalt aus einem Verzeichnis auf dem Remoteserver laden:

```
<script src="http://www.example.com/remote/remote.js"></script>
```

Die `sandboxRoot`-URL maskiert beliebigen Inhalt mit der gleichen URL auf dem Remoteserver. Im obigen Beispiel könnten Sie nicht auf Remoteinhalt unter `www.example.com/local/` (oder den zugehörigen Unterverzeichnissen) zugreifen, denn AIR ordnet die Anforderung dem lokalen Anwendungsverzeichnis neu zu. Anforderungen werden neu zugeordnet, ganz gleich, ob sie von der Seitennavigation, einem XMLHttpRequest oder einem anderen Verfahren zum Laden von Inhalt abgeleitet wurden.

Einrichten einer Schnittstelle für Sandbox-Brücken

Adobe AIR 1.0 und höher

Sie können eine Sandbox-Brücke verwenden, wenn Inhalt in der Anwendungs-Sandbox auf Eigenschaften oder Methoden zugreifen muss, die von Inhalt in einer anwendungsfremden Sandbox definiert wurden, oder wenn anwendungsfremder Inhalt auf Eigenschaften und Methoden zugreifen muss, die von Inhalt in der Anwendungs-Sandbox definiert wurden. Erstellen Sie eine Brücke mit der `childSandboxBridge`- und der `parentSandboxBridge`-Eigenschaft des `window`-Objekts eines beliebigen untergeordneten Dokuments.

Einrichten untergeordneter Sandbox-Brücken

Adobe AIR 1.0 und höher

Die `childSandboxBridge`-Eigenschaft ermöglicht dem untergeordneten Dokument die Offenlegung einer Schnittstelle für Inhalt im übergeordneten Dokument. Um eine Schnittstelle offen zu legen, legen Sie für die `childSandbox`-Eigenschaft eine Funktion oder ein Objekt im untergeordneten Dokument fest. Dann können Sie über Inhalt im übergeordneten Dokument auf das Objekt bzw. die Funktion zugreifen. Das folgende Beispiel zeigt, wie ein in einem untergeordneten Dokument ausgeführtes Skript ein Objekt, das eine Funktion und eine Eigenschaft enthält, für das übergeordnete Dokument offenlegen kann:

```
var interface = {};
interface.calculatePrice = function(){
    return ".45 cents";
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Wenn dieser untergeordnete Inhalt in einen Inlineframe mit der zugewiesenen ID „child“ geladen wurde, könnten Sie durch Lesen der `childSandboxBridge`-Eigenschaft des Frames über den übergeordneten Inhalt auf die Schnittstelle zugreifen:

```
var childInterface = document.getElementById("child").contentWindow.childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces ".45 cents"
air.trace(childInterface.storeID)); //traces "abc"
```

Einrichten übergeordneter Sandbox-Brücken

Adobe AIR 1.0 und höher

Die `parentSandboxBridge`-Eigenschaft ermöglicht dem übergeordneten Dokument die Offenlegung einer Schnittstelle für Inhalt im untergeordneten Dokument. Zum Offenlegen einer Schnittstelle legt das übergeordnete Dokument für die `parentSandbox`-Eigenschaft des untergeordneten Dokuments eine im übergeordneten Dokument definierte Funktion oder ein im übergeordneten Dokument definiertes Objekt fest. Dann können Sie über Inhalt im untergeordneten Dokument auf das Objekt bzw. die Funktion zugreifen. Das folgende Beispiel zeigt, wie ein im übergeordneten Frame ausgeführtes Skript ein Objekt offenlegen kann, das eine Funktion für ein untergeordnetes Dokument enthält:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").contentWindow.parentSandboxBridge = interface;
```

Mit dieser Schnittstelle könnte Inhalt im untergeordneten Frame Text in der Datei `save.txt` speichern, hätte aber keinen anderen Zugriff auf das Dateisystem. Der untergeordnete Inhalt könnte die save-Funktion wie folgt aufrufen:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Anwendungsinhalt sollte eine möglichst schmale Schnittstelle für andere Sandboxen offenlegen. Anwendungsfremder Inhalt sollte als von Natur aus nicht vertrauenswürdig angesehen werden, denn er könnte versehentlich mit Code oder mit bösartigem Code injiziert worden sein. Um einen Missbrauch der Schnittstelle, die über die übergeordnete Sandbox-Brücke offen gelegt wurde, zu verhindern, müssen entsprechende Sicherheitsmaßnahmen getroffen werden.

Zugreifen auf eine übergeordnete Sandbox-Brücke beim Laden einer Seite

Adobe AIR 1.0 und höher

Damit ein Skript in einem untergeordneten Dokument auf eine übergeordnete Sandbox-Brücke zugreifen kann, muss die Brücke eingerichtet werden, bevor das Skript ausgeführt wird. Fenster-, Frame- und Inlineframe-Objekte lösen nach der Erstellung eines neuen Seiten-DOM, aber vor der Analyse von Skripts oder dem Hinzufügen von DOM-Elementen ein `dominitialize`-Ereignis aus. Mit diesem Ereignis können Sie die Brücke so früh in der Seitenerstellungsfolge erstellen, dass alle Skripts im untergeordneten Dokument auf sie zugreifen können.

Das folgende Beispiel illustriert die Erstellung einer übergeordneten Sandbox-Brücke als Reaktion auf ein vom untergeordneten Frame ausgelöstes `dominitialize`-Ereignis:

```
<html>
<head>
<script>
var bridgeInterface = {};
bridgeInterface.testProperty = "Bridge engaged";
function engageBridge(){
    document.getElementById("sandbox").contentWindow.parentSandboxBridge = bridgeInterface;
}
</script>
</head>
<body>
<iframe id="sandbox"
            src="http://www.example.com/air/child.html"
            documentRoot="app:/"
            sandboxRoot="http://www.example.com/air/"
            ondominitialize="engageBridge()"/>
</body>
</html>
```

Das folgende `child.html`-Dokument verdeutlicht, wie untergeordnete Inhalte auf die übergeordnete Sandbox-Brücke zugreifen können:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Um statt in einem Frame in einem untergeordneten Fenster auf das `dominitialize`-Ereignis zu warten, müssen Sie den Listener dem neuen, von der `window.open()`-Funktion erstellten untergeordneten Fensterobjekt hinzufügen:

```
var childWindow = window.open();
childWindow.addEventListener("dominitialize", engageBridge());
childWindow.document.location = "http://www.example.com/air/child.html";
```

In diesem Fall ist es nicht möglich, Anwendungsinhalt einer anwendungsfremden Sandbox zuzuordnen. Dieses Verfahren ist nur sinnvoll, wenn `child.html` von außerhalb des Anwendungsverzeichnisses geladen wird. Sie können Anwendungsinhalt im Fenster auch weiterhin einer anwendungsfremden Sandbox zuordnen, müssen dazu allerdings zunächst eine Zwischenseite laden, die selbst Frames zum Laden des untergeordneten Dokuments verwendet, und es der gewünschten Sandbox zuordnen.

Wenn Sie mit der `createRootWindow()`-Funktion der HTMLLoader-Klasse ein Fenster erstellen, handelt es sich bei diesem nicht um ein untergeordnetes Fenster des Dokuments, aus dem `createRootWindow()` aufgerufen wird. Aus diesem Grund können Sie keine Sandbox-Brücke vom aufrufenden Fenster zu anwendungsfremdem Inhalt erstellen, der in das neue Fenster geladen wird. Stattdessen müssen Sie in dem neuen Fenster eine Zwischenseite laden, die zum Laden des untergeordneten Dokuments selbst Frames verwendet. Dann können Sie eine Brücke vom übergeordneten Dokument des neuen Fensters zum untergeordneten, in den Frame geladenen Dokument erstellen.

Kapitel 60: Skripterstellung von AIR-HTML-Containern

Adobe AIR 1.0 und höher

Die HTMLLoader-Klasse dient als Container für HTML-Inhalt in Adobe® AIR®. Mit den von der Klasse gebotenen Eigenschaften und Methoden, die von der Sprite-Klasse übernommen werden, können das Verhalten und das Erscheinungsbild des Objekts in der ActionScript™ 3.0-Anzeigeliste gesteuert werden. Außerdem werden von der Klasse Eigenschaften und Methoden definiert, mit denen beispielsweise Aufgaben wie das Laden von und Interagieren mit HTML-Inhalt sowie die Verlaufsverwaltung bewältigt werden können.

Die HTMLHost-Klasse definiert einen Satz an Standardverhaltensweisen für ein HTMLLoader-Objekt. Beim Erstellen eines HTMLLoader-Objekts wird keine HTMLHost-Implementierung bereitgestellt. Wenn also eines der Standardverhalten, wie etwa das Ändern der Fensterposition oder des Fenstertitels, durch den HTML-Inhalt ausgelöst wird, geschieht nichts. Sie können die HTMLHost-Klasse um Verhaltensdefinitionen erweitern, die Ihrer Anwendung entsprechen.

Für HTML-Fenster, die mit AIR erstellt wurden, wird eine Standardimplementierung von HTMLHost bereitgestellt. Sie können diese Standardimplementierung von HTMLHost einem anderen HTMLLoader-Objekt zuweisen, indem Sie die `htmlHost`-Eigenschaft des Objekts mithilfe eines neuen HTMLHost-Objekts festlegen, das mit dem `defaultBehavior`-Parameter gleich `true` erstellt wurde.

Hinweis: *Im Adobe® Flex™-Framework ist das HTMLLoader-Objekt in eine mx:HTML-Komponente eingebunden. Verwenden Sie mit Flex die HTML-Komponente.*

Anzeigeeigenschaften des HTMLLoader-Objekts

Adobe AIR 1.0 und höher

Ein HTMLLoader-Objekt übernimmt die Anzeigeeigenschaften der Adobe® Flash® Player Sprite-Klasse. Beispielsweise können Sie die Hintergrundfarbe verschieben, ausblenden und ändern sowie ihre Größe skalieren. Außerdem können Sie erweiterte Effekte wie etwa Filter, Masken, Skalierung und Drehung anwenden. Beachten Sie bei der Anwendung von Effekten, welche Auswirkung sie auf die Lesbarkeit haben. SWF- und PDF-Inhalt, der in eine HTML-Seite geladen ist, kann bei Anwendung gewisser Effekte nicht angezeigt werden.

HTML-Fenster enthalten ein HTMLLoader-Objekt, das den HTML-Inhalt darstellt. Dieses Objekt ist auf den Bereich innerhalb des Fensters beschränkt, sodass eine Änderung der Größe, der Position, der Drehung oder des Skalierungsfaktors manchmal zu unerwünschten Ergebnissen führt.

Grundlegende Anzeigeeigenschaften

Adobe AIR 1.0 und höher

Die grundlegenden Anzeigeeigenschaften von „HTMLLoader“ ermöglichen es, das Steuerelement im übergeordneten Anzeigeobjekt zu positionieren, die Größe einzustellen und das Steuerelement ein- oder auszublenden. Bei dem HTMLLoader-Objekt eines HTML-Fensters sollten Sie diese Eigenschaften nicht ändern.

Die grundlegenden Eigenschaften umfassen:

Eigenschaft	Hinweise
`x`, `y`	Positioniert das Objekt innerhalb des übergeordneten Containers.
`width`, `height`	Ändert die Maße des Anzeigebereichs.
`visible`	Steuert die Sichtbarkeit des Objekts und des darin enthaltenen Inhalts.

Außerhalb eines HTML-Fensters werden die `width`- und `height`-Eigenschaften eines HTMLLoader-Objekts auf den Standardwert „0“ eingestellt. Sie müssen die Breite und Höhe einstellen, damit der geladene HTML-Inhalt angezeigt werden kann. Der HTML-Inhalt wird gemäß der HTMLLoader-Größe gezeichnet und gemäß der im Inhalt festgelegten HTML- und CSS-Eigenschaften angeordnet. Durch eine Änderung der HTMLLoader-Größe fließt der Inhalt neu ein.

Beim Laden von Inhalt in ein neues HTMLLoader-Objekt (wobei der Wert für `width` noch „0“ lautet) könnten Sie dazu neigen, `width` und `height` von „HTMLLoader“ mithilfe der `contentWidth`- und `contentHeight`-Eigenschaften einzustellen. Diese Technik funktioniert bei Seiten, die bei der Anordnung gemäß der HTML- und CSS-Flussregeln eine angemessene Mindestbreite aufweisen. Wird von „HTMLLoader“ jedoch keine angemessene Breite angegeben, fließen manche Seiten in ein langes und schmales Layout.

Hinweis: *Wenn Sie die width- und height-Werte eines HTMLLoader-Objekts ändern, werden die scaleX- und scaleY-Werte nicht geändert (wie es bei den meisten anderen Anzeigeobjekttypen der Fall wäre).*

Transparenz von HTMLLoader-Inhalt

Adobe AIR 1.0 und höher

Mit der `paintsDefaultBackground`-Eigenschaft eines HTMLLoader-Objekts, deren Standardwert `true` lautet, wird bestimmt, ob das HTMLLoader-Objekt einen opaken Hintergrund zeichnet. Wenn `paintsDefaultBackground` den Wert `false` aufweist, ist der Hintergrund transparent. Der Anzeigeobjektcontainer oder andere dem HTMLLoader-Objekt untergeordnete Anzeigeobjekte sind hinter den Vordergrundelementen des HTML-Inhalts sichtbar.

Wenn das body-Element oder ein anderes Element des HTML-Dokuments eine Hintergrundfarbe angibt (z. B. mit `style="background-color:gray"`), ist der Hintergrund dieses HTML-Teils opak und wird mit der angegebenen Hintergrundfarbe wiedergegeben. Wenn Sie die `opaqueBackground`-Eigenschaft des HTMLLoader-Objekts einstellen und `paintsDefaultBackground` lautet `false`, ist die für `opaqueBackground` eingestellte Farbe sichtbar.

Hinweis: *Mithilfe einer PNG-Grafik können Sie einen Hintergrund mit Alpha-Mischung für ein Element in einem HTML-Dokument bereitstellen. Die Einstellung des Opazitätsstils eines HTML-Elements wird nicht unterstützt.*

Skalierung von HTMLLoader-Inhalt

Adobe AIR 1.0 und höher

Vermeiden Sie es, ein HTMLLoader-Objekt mit einem Skalierungsfaktor von über 1,0 zu skalieren. Der Text im HTMLLoader-Inhalt wird anhand einer bestimmten Auflösung wiedergegeben und bei einer Vergrößerung des HTMLLoader-Objekts sind einzelne Pixelpunkte zu sehen. Um zu verhindern, dass das HTMLLoader-Objekt sowie sein Inhalt skaliert werden, wenn die Größe eines Fensters geändert wird, stellen Sie die `scaleMode`-Eigenschaft der Bühne auf `StageScaleMode.NO_SCALE` ein.

Erwägungen beim Laden von SWF- oder PDF-Inhalt in eine HTML-Seite

Adobe AIR 1.0 und höher

In ein HTMLLoader-Objekt geladener SWF- und PDF-Inhalt wird unter folgenden Bedingungen nicht angezeigt:

- Das HTMLLoader-Objekt wird mit einem Faktor von über 1,0 skaliert.
- Die alpha-Eigenschaft des HTMLLoader-Objekts wird auf einen anderen Wert als 1,0 eingestellt.
- Der HTMLLoader-Inhalt wird gedreht.

Der Inhalt wird wieder angezeigt, wenn Sie die widersprüchliche Eigenschaftseinstellung und die aktiven Filter entfernen.

Außerdem kann PDF-Inhalt in der Laufzeitumgebung nicht in transparenten Fenstern angezeigt werden. In der Laufzeit wird SWF-Inhalt, der in einer HTML-Seite eingebettet ist, nur angezeigt, wenn der `wmode`-Parameter des object- oder embed-Tags auf `opaque` oder `transparent` eingestellt ist. Da `wmode` den Standardwert `window` hat, wird SWF-Inhalt in transparenten Fenstern nur angezeigt, wenn Sie den wmode-Parameter explizit festlegen.

Hinweis: *Vor AIR 1.5.2 konnte in HTML eingebetteter SWF-Inhalt grundsätzlich nicht angezeigt werden, unabhängig vom wmode-Wert.*

Weitere Informationen zum Laden dieser Medientypen in einen HTMLLoader finden Sie unter „Einbetten von SWF-Inhalten in HTML“ auf Seite 1060 und „Hinzufügen von PDF-Inhalten in AIR“ auf Seite 587.

Erweiterte Anzeigeeigenschaften

Adobe AIR 1.0 und höher

Die HTMLLoader-Klasse übernimmt mehrere Methoden, die für Spezialeffekte verwendet werden können. Im Allgemeinen weisen diese Effekte bei der Verwendung mit der HTMLLoader-Anzeige gewisse Beschränkungen auf, aber sie können bei Übergängen oder anderen temporären Effekten nützlich sein. Bei der Anzeige eines Dialogfelds, in dem Benutzer Eingaben vornehmen, könnte beispielsweise das Hauptfenster unscharf angezeigt werden, bis der Benutzer das Dialogfeld schließt. Entsprechend könnte beim Schließen des Fensters die Anzeige ausgeblendet werden.

Die erweiterten Anzeigeeigenschaften umfassen:

Eigenschaft	Einschränkungen
`alpha`	Kann die Lesbarkeit von HTML-Inhalt verschlechtern.
`filters`	In einem HTML-Fenster werden äußere Effekte von der Fensterkante abgeschnitten.
`graphics`	Mithilfe von Grafikbefehlen gezeichnete Formen werden unter dem HTML-Inhalt, einschließlich dem Standardhintergrund, angezeigt. Der Wert der paintsDefaultBackground-Eigenschaft muss „false“ lauten, damit die gezeichneten Formen sichtbar sind.
`opaqueBackground`	Ändert nicht die Farbe des Standardhintergrunds. Der Wert der paintsDefaultBackground-Eigenschaft muss „false“ lauten, damit diese Farbebene sichtbar ist.

Eigenschaft	Einschränkungen
`rotation`	Die Ecken des rechteckigen HTMLLoader-Bereichs werden möglicherweise von der Fensterkante beschnitten. In den HTML-Inhalt geladener SWF- und PDF-Inhalt wird nicht angezeigt.
`scaleX`, `scaleY`	Bei der Wiedergabe mit Skalierungsfaktoren von über 1 können einzelne Pixelpunkte zu sehen sein. In den HTML-Inhalt geladener SWF- und PDF-Inhalt wird nicht angezeigt.
`transform`	Kann die Lesbarkeit von HTML-Inhalt verschlechtern. Die HTML-Anzeige wird möglicherweise von der Fensterkante beschnitten. In den HTML-Inhalt geladener SWF- und PDF-Inhalt wird nicht angezeigt, wenn die Transformierung eine Drehung, Skalierung oder Neigung umfasst.

Das folgende Beispiel zeigt, wie das `filters`-Array eingestellt werden muss, um die gesamte HTML-Anzeige unscharf anzuzeigen:

```
 var html:HTMLLoader = new HTMLLoader();
var urlReq:URLRequest = new URLRequest("http://www.adobe.com/");
html.load(urlReq);
html.width = 800;
html.height = 600;

var blur:BlurFilter = new BlurFilter(8);
var filters:Array = [blur];
html.filters = filters;
```

Bildlauf von HTML-Inhalt

Adobe AIR 1.0 und höher

Die HTMLLoader-Klasse bietet die folgenden Eigenschaften, mit denen Sie den Bildlauf von HTML-Inhalt steuern können:

Eigenschaft	Beschreibung
`contentHeight`	Die Höhe des HTML-Inhalts in Pixel.
contentWidth	Die Breite des HTML-Inhalts in Pixel.
scrollH	Die horizontale Bildlaufposition des HTML-Inhalts innerhalb des HTMLLoader-Objekts.
scrollV	Die vertikale Bildlaufposition des HTML-Inhalts innerhalb des HTMLLoader-Objekts.

Im folgenden Beispiel wird die `scrollV`-Eigenschaft so eingestellt, dass am HTML-Inhalt ein Bildlauf zum Seitenende durchgeführt wird:

```
 var html:HTMLLoader = new HTMLLoader();
html.addEventListener(Event.HTML_BOUNDS_CHANGE, scrollHTML);

const SIZE:Number = 600;
html.width = SIZE;
html.height = SIZE;

var urlReq:URLRequest = new URLRequest("http://www.adobe.com");
html.load(urlReq);
this.addChild(html);

function scrollHTML(event:Event):void
{
    html.scrollV = html.contentHeight - SIZE;
}
```

Der „HTMLLoader“ enthält keine horizontalen und vertikalen Bildlaufleisten. Bildlaufleisten können Sie in ActionScript oder mithilfe einer Flex-Komponente implementieren. Die Flex HTML-Komponente umfasst automatisch Bildlaufleisten für HTML-Inhalt. Außerdem können Sie mithilfe der `HTMLLoader.createRootWindow()`-Methode ein Fenster erstellen, das ein HTMLLoader-Objekt mit Bildlaufleisten enthält (siehe „Erstellen von Fenstern mit HTML-Inhalt und Bildlauf“ auf Seite 1088).

Zugreifen auf die HTML-Verlaufsliste

Adobe AIR 1.0 und höher

Neue Seiten, die in ein HTMLLoader-Objekt geladen werden, werden von der Laufzeitumgebung in einer Verlaufsliste für das Objekt verzeichnet. Die Verlaufsliste entspricht dem `window.history`-Objekt der HTML-Seite. Die HTMLLoader-Klasse bietet die folgenden Eigenschaften und Methoden, die Ihnen die Arbeit mit der HTML-Verlaufsliste ermöglichen:

Klassenmitglied	Beschreibung
`historyLength`	Die Gesamtlänge der Verlaufsliste, einschließlich der Zurück- und Vorwärts-Einträge.
`historyPosition`	Die aktuelle Position in der Verlaufsliste. Verlaufselemente vor dieser Position stellen eine Zurück-Navigation dar und Elemente nach dieser Position stellen eine Vorwärts-Navigation dar.
`getHistoryAt()`	Gibt das URLRequest-Objekt zurück, das dem Verlaufseintrag an der angegebenen Position in der Verlaufsliste entspricht.
`historyBack()`	Navigiert in der Verlaufsliste zurück, sofern möglich.
`historyForward()`	Navigiert in der Verlaufsliste vorwärts, sofern möglich.
`historyGo()`	Navigiert die angegebene Anzahl von Schritten im Browserverlauf. Navigiert vorwärts bei positiven Werten, zurück bei negativen. Bei Navigieren zu null wird die Seite neu geladen. Wird eine Position nach dem Ende angegeben, wird an das Ende der Liste navigiert.

Elemente in der Verlaufsliste werden als Objekte mit dem HTMLHistoryItem-Typ gespeichert. Die HTMLHistoryItem-Klasse hat die folgenden Eigenschaften:

Eigenschaft	Beschreibung
`isPost`	Auf `true` einstellen, wenn die HTML-Seite POST-Daten enthält.
`originalUrl`	Die ursprüngliche URL der HTML-Seite vor allen Umleitungen.
`title`	Der Titel der HTML-Seite.
`url`	Die URL der HTML-Seite.

Festlegen des Benutzer-Agent-Strings, der beim Laden von HTML-Inhalt verwendet wird

Adobe AIR 1.0 und höher

Die HTMLLoader-Klasse weist eine `userAgent`-Eigenschaft auf, mit der Sie den vom „HTMLLoader“ verwendeten Benutzer-Agent-String festlegen können. Legen Sie die `userAgent`-Eigenschaft des HTMLLoader-Objekts fest, bevor Sie die `load()`-Methode aufrufen. Wenn Sie diese Eigenschaft der HTMLLoader-Instanz festlegen, wird die `userAgent`-Eigenschaft der an die `load()`-Methode übergebenen URLRequest *nicht* verwendet.

Sie können den Benutzer-Agent-Standardstring festlegen, der von allen HTMLLoader-Objekten in einer Anwendungsdomäne verwendet wird, indem Sie die `URLRequestDefaults.userAgent`-Eigenschaft einstellen. Die statischen URLRequestDefaults-Eigenschaften gelten als Standardwerte für alle URLRequest-Objekte, nicht nur für die URLRequest-Objekte, die mit der `load()`-Methode von HTMLLoader-Objekten verwendet werden. Das Festlegen der `userAgent`-Eigenschaft eines HTMLLoader-Objekts setzt die `URLRequestDefaults.userAgent`-Standardeinstellung außer Kraft.

Wenn Sie einen Benutzer-Agent-Wert weder für die `userAgent`-Eigenschaft des HTMLLoader-Objekts noch für `URLRequestDefaults.userAgent` festlegen, wird der AIR-Benutzer-Agent-Standardwert verwendet. Dieser Standardwert variiert je nach Laufzeit-Betriebssystem (z. B. Mac OS oder Windows), der Laufzeitsprache und der Laufzeitversion wie in den folgenden beiden Beispielen:

- `"Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/420+ (KHTML, wie Gecko) AdobeAIR/1.0"`
- `"Mozilla/5.0 (Windows; U; en) AppleWebKit/420+ (KHTML, wie Gecko) AdobeAIR/1.0"`

Festlegen der mit HTML-Inhalt verwendeten Zeichenkodierung

Adobe AIR 1.0 und höher

Von einer HTML-Seite kann durch Einschließen eines `meta`-Tags (siehe folgendes Beispiel) die verwendete Zeichenkodierung angegeben werden:

```
meta http-equiv="content-type" content="text/html" charset="ISO-8859-1";
```

Sie können die Seiteneinstellung außer Kraft setzen, um eine bestimmte Zeichenkodierung zu verwenden, indem Sie die `textEncodingOverride`-Eigenschaft des HTMLLoader-Objekts festlegen:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingOverride = "ISO-8859-1";
```

Sie können eine Zeichenkodierung für den HTMLLoader-Inhalt festlegen, wenn von einer HTML-Seite die `textEncodingFallback`-Eigenschaft des HTMLLoader-Objekts nicht festgelegt wird:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingFallback = "ISO-8859-1";
```

Die `textEncodingOverride`-Eigenschaft setzt die Einstellung in der HTML-Seite außer Kraft. Und die `textEncodingOverride`-Eigenschaft und die Einstellung in der HTML-Seite setzen die `textEncodingFallback`-Eigenschaft außer Kraft.

Legen Sie die `textEncodingOverride`-Eigenschaft oder die `textEncodingFallback`-Eigenschaft fest, bevor der HTML-Inhalt geladen wird.

Definieren browserähnlicher Benutzeroberflächen für HTML-Inhalt

Adobe AIR 1.0 und höher

JavaScript bietet mehrere APIs zur Steuerung des Fensters, in dem der HTML-Inhalt angezeigt wird. In AIR können diese APIs durch die Implementierung einer benutzerdefinierten HTMLHost-Klasse außer Kraft gesetzt werden.

Erweitern der HTMLHost-Klasse

Adobe AIR 1.0 und höher

Wenn Ihre Anwendung beispielsweise mehrere HTMLLoader-Objekte in einer Benutzeroberfläche mit Registerkarten darstellt, sollten durch die von den geladenen HTML-Seiten vorgenommenen Titeländerungen nicht der Titel des Hauptfensters sondern die Bezeichnung der Registerkarte geändert werden. In ähnlicher Weise könnte der Code auf einen Aufruf von `window.moveTo()` reagieren und die Position des HTMLLoader-Objekts im übergeordneten Anzeigeobjektcontainer ändern, das HTMLLoader-Objekt enthaltende Fenster verschieben, nichts unternehmen oder etwas vollkommen anders ausführen.

Mit der AIR-HTMLHost-Klasse werden die folgenden JavaScript-Eigenschaften und -Methoden gesteuert:

- `window.status`
- `window.document.title`
- `window.location`
- `window.blur()`
- `window.close()`
- `window.focus()`
- `window.moveBy()`
- `window.moveTo()`
- `window.open()`
- `window.resizeBy()`

- `window.resizeTo()`

Wenn Sie ein HTMLLoader-Objekt mithilfe von `new HTMLLoader()` erstellen, sind die genannten JavaScript-Eigenschaften und -Methoden nicht aktiviert. Die HTMLHost-Klasse bietet eine browserähnliche Standardimplementierung dieser JavaScript-APIs. Außerdem können Sie die HTMLHost-Klasse erweitern, um das Verhalten anzupassen. Um ein HTMLHost-Objekt zu erstellen, das das Standardverhalten unterstützt, müssen Sie im HTMLHost-Konstruktor den `defaultBehaviors`-Parameter auf „true“ festlegen:

```
var defaultHost:HTMLHost = new HTMLHost(true);
```

Wenn Sie in AIR ein HTML-Fenster mithilfe der `createRootWindow()`-Methode der HTMLLoader-Klasse erstellen, wird automatisch eine HTMLHost-Instanz, die das Standardverhalten unterstützt, zugewiesen. Sie können das Verhalten des Host-Objekts ändern, indem Sie der `htmlHost`-Eigenschaft von „HTMLLoader“ eine andere HTMLHost-Implementierung zuweisen, oder Sie können die Funktionen durch Zuweisen von `null` vollständig deaktivieren.

Hinweis: *Dem ersten für eine HTML-basierte AIR-Anwendung erstellten Fenster und jedem Fenster, das durch die Standardimplementierung der JavaScript-Methode `window.open()` erstellt wird, wird von AIR ein HTMLHost-Standardobjekt zugewiesen.*

Beispiel: Erweitern der HTMLHost-Klasse

Adobe AIR 1.0 und höher

Das folgende Beispiel zeigt, wie Sie die Wirkungsweise eines HTMLLoader-Objekts auf die Benutzeroberfläche durch das Erweitern der HTMLHost-Klasse anpassen können:

Flex-Beispiel:

1. Erstellen Sie eine Klasse, die die HTMLHost-Klasse erweitert (eine Unterklasse).
2. Setzen Sie Methoden der neuen Klasse außer Kraft, um Änderungen der Einstellungen in Bezug auf die Benutzeroberfläche zu verarbeiten. Beispielsweise definiert die CustomHost-Klasse Verhalten bei Aufrufen von `window.open()` und Änderungen an `window.document.title`. Durch Aufrufe von `window.open()` wird die HTML-Seite in einem neuen Fenster geöffnet und durch Änderungen an `window.document.title` (einschließlich der Festlegung des `<title>`-Elements einer HTML-Seite) wird der Titel dieses Fensters festgelegt.

```
package
{
    import flash.html.*;
    import flash.display.StageScaleMode;
    import flash.display.NativeWindow;
    import flash.display.NativeWindowInitOptions;

    public class CustomHost extends HTMLHost
    {
        import flash.html.*;
        override public function
            createWindow(windowCreateOptions:HTMLWindowCreateOptions):HTMLLoader
        {
            var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
            var bounds:Rectangle = new Rectangle(windowCreateOptions.x,
                                                 windowCreateOptions.y,
                                                 windowCreateOptions.width,
                                                 windowCreateOptions.height);
            var htmlControl:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions,
                                         windowCreateOptions.scrollBarsVisible, bounds);
            htmlControl.htmlHost = new HTMLHostImplementation();
            if(windowCreateOptions.fullscreen){
                htmlControl.stage.displayState =
                    StageDisplayState.FULL_SCREEN_INTERACTIVE;
            }
            return htmlControl;
        }
        override public function updateTitle(title:String):void
        {
            htmlLoader.stage.nativeWindow.title = title;
        }
    }
}
```

3 Erstellen Sie ein Objekt der neuen Klasse im Code, der „HTMLLoader“ enthält (nicht im Code der neuen Unterklasse von „HTMLHost“). Weisen Sie das neue Objekt der `htmlHost`-Eigenschaft von „HTMLLoaders“ zu. Im folgenden Flex-Code wird die im vorherigen Schritt definierte CustomHost-Klasse verwendet:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication
    xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    applicationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.html.HTMLLoader;
            import CustomHost;
            private function init():void
            {
                var html:HTMLLoader = new HTMLLoader();
                html.width = container.width;
                html.height = container.height;
                var urlReq:URLRequest = new URLRequest("Test.html");
                html.htmlHost = new CustomHost();
                html.load(urlReq);
                container.addChild(html);
            }
        ]]>
    </mx:Script>
    <mx:UIComponent id="container" width="100%" height="100%"/>
</mx:WindowedApplication>
```

Um den hier beschriebenen Code zu prüfen, schließen Sie eine HTML-Datei mit dem folgenden Inhalt in das Anwendungsverzeichnis ein:

```
<html>
    <head>
        <title>Test</title>
    </head>
    <script>
        function openWindow()
        {
            window.runtime.trace("in");
            document.title = "foo"
            window.open('Test.html');
            window.runtime.trace("out");
        }
    </script>
    <body>
        <a href="#" onclick="openWindow()">window.open('Test.html')</a>
    </body>
</html>
```

Flash Professional-Beispiel:

1 Erstellen Sie eine Flash-Datei für AIR. Legen Sie die Dokumentklasse „CustomHostExample“ fest und speichern Sie die Datei dann unter „CustomHostExample.fla“.

2 Erstellen Sie eine ActionScript-Datei namens „CustomHost.as“, die eine Klasse enthält, die die HTMLHost-Klasse erweitert (eine Unterklasse). Diese Klasse setzt gewisse Methoden der neuen Klasse außer Kraft, um Änderungen der Einstellungen in Bezug auf die Benutzeroberfläche zu verarbeiten. Beispielsweise definiert die CustomHost-Klasse Verhalten bei Aufrufen von `window.open()` und Änderungen an `window.document.title`. Durch Aufrufe der `window.open()`-Methode wird die HTML-Seite in einem neuen Fenster geöffnet und durch Änderungen an der `window.document.title`-Eigenschaft (einschließlich der Festlegung des `<title>`-Elements einer HTML-Seite) wird der Titel dieses Fensters festgelegt.

```
package
{
    import flash.display.StageScaleMode;
    import flash.display.NativeWindow;
    import flash.display.NativeWindowInitOptions;
    import flash.events.Event;
    import flash.events.NativeWindowBoundsEvent;
    import flash.geom.Rectangle;
    import flash.html.HTMLLoader;
    import flash.html.HTMLHost;
    import flash.html.HTMLWindowCreateOptions;
    import flash.text.TextField;

    public class CustomHost extends HTMLHost
    {
        public var statusField:TextField;

        public function CustomHost(defaultBehaviors:Boolean=true)
        {
            super(defaultBehaviors);
        }
        override public function windowClose():void
        {
            htmlLoader.stage.nativeWindow.close();
        }
        override public function createWindow(
                                windowCreateOptions:HTMLWindowCreateOptions ):HTMLLoader
        {
            var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
            var bounds:Rectangle = new Rectangle(windowCreateOptions.x,
                                        windowCreateOptions.y,
                                        windowCreateOptions.width,
                                        windowCreateOptions.height);
            var htmlControl:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions,
                                    windowCreateOptions.scrollBarsVisible, bounds);
            htmlControl.htmlHost = new HTMLHostImplementation();
            if(windowCreateOptions.fullscreen){
                htmlControl.stage.displayState =
                    StageDisplayState.FULL_SCREEN_INTERACTIVE;
            }
            return htmlControl;
        }
        override public function updateLocation(locationURL:String):void
        {
            trace(locationURL);
        }
        override public function set windowRect(value:Rectangle):void
        {
            htmlLoader.stage.nativeWindow.bounds = value;
```

```
        }
        override public function updateStatus(status:String):void
        {
            statusField.text = status;
            trace(status);
        }
        override public function updateTitle(title:String):void
        {
            htmlLoader.stage.nativeWindow.title = title + "- Example Application";
        }
        override public function windowBlur():void
        {
            htmlLoader.alpha = 0.5;
        }
        override public function windowFocus():void
        {
            htmlLoader.alpha = 1;
        }
    }
}
```

3 Erstellen Sie eine weitere ActionScript-Datei namens „CustomHostExample.as“, die die Dokumentklasse der Anwendung enthält. Diese Klasse erstellt ein HTMLLoader-Objekt und legt für die host-Eigenschaft eine Instanz der im vorherigen Schritt definierten CustomHost-Klasse fest:

```
package
{
    import flash.display.Sprite;
    import flash.html.HTMLLoader;
    import flash.net.URLRequest;
    import flash.text.TextField;

    public class CustomHostExample extends Sprite
    {
        function CustomHostExample():void
        {
            var html:HTMLLoader = new HTMLLoader();
            html.width = 550;
            html.height = 380;
            var host:CustomHost = new CustomHost();
            html.htmlHost = host;

            var urlReq:URLRequest = new URLRequest("Test.html");
            html.load(urlReq);

            addChild(html);

            var statusTxt:TextField = new TextField();
            statusTxt.y = 380;
            statusTxt.height = 20;
            statusTxt.width = 550;
            statusTxt.background = true;
            statusTxt.backgroundColor = 0xEEEEEEEE;
            addChild(statusTxt);

            host.statusField = statusTxt;
        }
    }
}
```

Um den hier beschriebenen Code zu prüfen, schließen Sie eine HTML-Datei mit dem folgenden Inhalt in das Anwendungsverzeichnis ein:

```
<html>
    <head>
    <title>Test</title>
    <script>
    function openWindow()
    {
    document.title = "Test"
    window.open('Test.html');
    }
    </script>
    </head>
    <body bgColor="#EEEEEE">
    <a href="#" onclick="window.open('Test.html')">window.open('Test.html')</a>
    <br/><a href="#" onclick="window.document.location='http://www.adobe.com'">
    window.document.location = 'http://www.adobe.com'</a>
    <br/><a href="#" onclick="window.moveBy(6, 12)">moveBy(6, 12)</a>
    <br/><a href="#" onclick="window.close()">window.close()</a>
    <br/><a href="#" onclick="window.blur()">window.blur()</a>
    <br/><a href="#" onclick="window.focus()">window.focus()</a>
    <br/><a href="#" onclick="window.status = new Date().toString()">window.status=new
Date().toString()</a>
    </body>
</html>
```

Verarbeiten von Änderungen an der window.location-Eigenschaft

Adobe AIR 1.0 und höher

Setzen Sie die `locationChange()`-Methode außer Kraft, um Änderungen an der URL der HTML-Seite zu verarbeiten. Die `locationChange()`-Methode wird aufgerufen, wenn der Wert von `window.location` durch JavaScript in einer Seite geändert wird. Im folgenden Beispiel wird einfach die angeforderte URL geladen:

```
override public function updateLocation(locationURL:String):void
{
    htmlLoader.load(new URLRequest(locationURL));
}
```

Hinweis: *Mit der htmlLoader-Eigenschaft des HTMLHost-Objekts können Sie auf das aktuelle HTMLLoader-Objekt verweisen.*

Verarbeiten von JavaScript-Aufrufen von „window.moveBy()“, „window.moveTo()“, „window.resizeTo()“, „window.resizeBy()“

Adobe AIR 1.0 und höher

Setzen Sie die `set windowRect()`-Methode außer Kraft, um Änderungen an den Begrenzungen des HTML-Inhalts zu verarbeiten. Die `set windowRect()`-Methode wird aufgerufen, wenn `window.moveBy()`, `window.moveTo()`, `window.resizeTo()` oder `window.resizeBy()` durch das JavaScript einer Seite aufgerufen wird. Im folgenden Beispiel werden einfach die Begrenzungen des Desktop-Fensters aktualisiert:

```
override public function set windowRect(value:Rectangle):void
{
    htmlLoader.stage.nativeWindow.bounds = value;
}
```

Verarbeiten von JavaScript-Aufrufen von „window.open()“

Adobe AIR 1.0 und höher

Setzen Sie die `createWindow()`-Methode außer Kraft, um Aufrufe von `window.open()` durch JavaScript zu verarbeiten. Implementierungen der `createWindow()`-Methode sorgen für das Erstellen und Zurückgeben eines neuen HTMLLoader-Objekts. „HTMLLoader“ wird in der Regel in einem neuen Fenster angezeigt, aber es ist nicht erforderlich, ein neues Fenster zu erstellen.

Das folgende Beispiel illustriert, wie die `createWindow()`-Funktion mithilfe von `HTMLLoader.createRootWindow()` implementiert wird, um sowohl das Fenster als auch das HTMLLoader-Objekt zu erstellen. Alternativ können Sie auch separat ein NativeWindow-Objekt erstellen und „HTMLLoader“ zur Fensterbühne hinzufügen.

```
override public function createWindow(windowCreateOptions:HTMLWindowCreateOptions):HTMLLoader{
    var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
    var bounds:Rectangle = new Rectangle(windowCreateOptions.x, windowCreateOptions.y,
                                 windowCreateOptions.width, windowCreateOptions.height);
    var htmlControl:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions,
                                     windowCreateOptions.scrollBarsVisible, bounds);
    htmlControl.htmlHost = new HTMLHostImplementation();
    if(windowCreateOptions.fullscreen){
        htmlControl.stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
    }
    return htmlControl;
}
```

Hinweis: *In diesem Beispiel wird die benutzerdefinierte HTMLHost-Implementierung jedem neuen Fenster, das mit `window.open()` erstellt wird, zugewiesen. Alternativ können Sie auch eine andere Implementierung verwenden oder die `htmlHost`-Eigenschaft für neue Fenster auf „null“ setzen.*

Das Objekt, das als Parameter an die `createWindow()`-Methode übergeben wird, ist ein HTMLWindowCreateOptions-Objekt. Die HTMLWindowCreateOptions-Klasse enthält Eigenschaften, die die im `features`-Parameterstring im Aufruf von `window.open()` festgelegten Werte melden:

HTMLWindowCreateOptions -Eigenschaft	Entsprechende Einstellung im Funktionsstring im JavaScript-Aufruf von „window.open()“
`fullscreen`	`fullscreen`
`height`	`height`
`locationBarVisible`	`location`
`menuBarVisible`	`menubar`
`resizeable`	`resizable`
`scrollBarsVisible`	`scrollbars`
`statusBarVisible`	`status`
`toolBarVisible`	`toolbar`
`width`	`width`
`x`	`left` oder `screenX`
`y`	`top` oder `screenY`

Von der HTMLLoader-Klasse werden nicht alle Funktionen implementiert, die im Funktionsstring angegeben werden können. Die Anwendung muss bei Bedarf Bildlaufleisten, Adressleisten, Menüleisten, Statusleisten und Symbolleisten bieten.

Die anderen Argumente der `window.open()`-Methode von JavaScript werden vom System verarbeitet. Mit einer `createWindow()`-Implementierung sollte kein Inhalt in das HTMLLoader-Objekt geladen oder der Fenstertitel festgelegt werden.

Verarbeiten von JavaScript-Aufrufen von „window.close()“

Adobe AIR 1.0 und höher

Setzen Sie die `windowClose()`-Methode außer Kraft, um Aufrufe der `window.close()`-Methode durch JavaScript zu verarbeiten. Im folgenden Beispiel wird das Desktop-Fenster geschlossen, wenn die `window.close()`-Methode aufgerufen wird:

```
override public function windowClose():void
{
    htmlLoader.stage.nativeWindow.close();
}
```

Bei JavaScript-Aufrufen von `window.close()` muss das enthaltende Fenster nicht geschlossen werden. Im folgenden Beispiel wird gezeigt, wie Sie „HTMLLoader“ aus der Anzeigeliste entfernen und das Fenster (das weiteren Inhalt enthalten kann) offen lassen:

```
override public function windowClose():void
{
    htmlLoader.parent.removeChild(htmlLoader);
}
```

Verarbeiten von Änderungen an der window.status-Eigenschaft

Adobe AIR 1.0 und höher

Setzen Sie die `updateStatus()`-Methode außer Kraft, um Änderungen am Wert von `window.status` durch JavaScript zu verarbeiten. In diesem Beispiel wird der Statuswert ermittelt:

```
 override public function updateStatus(status:String):void
{
    trace(status);
}
```

Der angeforderte Status wird als String an die `updateStatus()`-Methode übergeben.

Das HTMLLoader-Objekt bietet keine Statusleiste.

Verarbeiten von Änderungen an der window.document.title-Eigenschaft

Adobe AIR 1.0 und höher

Setzen Sie die `updateTitle()`-Methode außer Kraft, um Änderungen am Wert von `window.document.title` durch JavaScript zu verarbeiten. Im folgenden Beispiel wird der Fenstertitel geändert und der String "Sample" wird an den Titel angehängt:

```
override public function updateTitle(title:String):void
{
    htmlLoader.stage.nativeWindow.title = title + " - Sample";
}
```

Wenn `document.title` auf einer HTML-Seite festgelegt ist, wird der angeforderte Titel als String an die `updateTitle()`-Methode übergeben.

Durch Änderungen an `document.title` muss nicht unbedingt auch der Titel des Fensters geändert werden, in dem das HTMLLoader-Objekt enthalten ist. Sie können beispielsweise ein anderes Element der Benutzeroberfläche, wie etwa ein Textfeld, ändern.

Verarbeiten von JavaScript-Aufrufen von „window.blur()“ und „window.focus()“

Adobe AIR 1.0 und höher

Setzen Sie die `windowBlur()`- und `windowFocus()`-Methoden außer Kraft, um Aufrufe der `window.blur()`- und `window.focus()`-Methoden durch JavaScript zu verarbeiten (siehe folgendes Beispiel):

```
override public function windowBlur():void
{
    htmlLoader.alpha = 0.5;
}
override public function windowFocus():void
{
    htmlLoader.alpha = 1.0;
    NativeApplication.nativeApplication.activate(htmlLoader.stage.nativeWindow);
}
```

Hinweis: *AIR bietet keine API zum Deaktivieren eines Fensters oder einer Anwendung.*

Erstellen von Fenstern mit HTML-Inhalt und Bildlauf

Adobe AIR 1.0 und höher

Die HTMLLoader-Klasse enthält die statische Methode `HTMLLoader.createRootWindow()`. Diese Methode ermöglicht es, ein neues Fenster (durch ein NativeWindow-Objekt dargestellt) zu öffnen, das ein HTMLLoader-Objekt enthält. Außerdem können Sie einige Benutzeroberflächeneinstellungen für das Fenster festlegen. Mithilfe von vier Parametern der Methode können Sie die Benutzerschnittstelle definieren:

Parameter	Beschreibung
`visible`	Ein boolescher Wert, der angibt, ob das Fenster zunächst sichtbar (`true`) oder nicht sichtbar (`false`) ist.
`windowInitOptions`	Ein NativeWindowInitOptions-Objekt. Mit der NativeWindowInitOptions-Klasse werden Initialisierungsoptionen für ein NativeWindow-Objekt definiert, u. a.: ob das Fenster minimiert oder maximiert oder ob seine Größe geändert werden kann, ob das Fenster über System-Fensterdesign oder benutzerdefiniertes Fensterdesign verfügt, ob das Fenster transparent ist oder nicht (bei Fenstern, die kein System-Fensterdesign bieten) und der Fenstertyp.
`scrollBarsVisible`	Ob Bildlaufleisten vorhanden sind (`true`) oder nicht (`false`).
`bounds`	Ein Rectangle-Objekt, das die Position und Größe des neuen Fensters definiert.

Im folgenden Beispiel wird mithilfe der `HTMLLoader.createRootWindow()`-Methode ein Fenster mit HTMLLoader-Inhalt und Bildlaufleisten erstellt:

```
var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
var bounds:Rectangle = new Rectangle(10, 10, 600, 400);
var html2:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions, true, bounds);
var urlReq2:URLRequest = new URLRequest("http://www.example.com");
html2.load(urlReq2);
html2.stage.nativeWindow.activate();
```

Hinweis: *Fenster, die durch Aufrufen von `createRootWindow()` direkt in JavaScript erstellt werden, bleiben unabhängig vom öffnenden HTML-Fenster. Beispielsweise lautet der Wert der Fenstereigenschaften `opener` und `parent` von JavaScript `null`. Wenn Sie `createRootWindow()` jedoch indirekt aufrufen, indem Sie die `createWindow()`-Methode von „HTMLHost" außer Kraft setzen, um `createRootWindow()` aufzurufen, verweisen `opener` und `parent` auf das öffnende HTML-Fenster.*

Erstellen von Unterklassen für die HTMLLoader-Klasse

Adobe AIR 1.0 und höher

Sie können eine Unterklasse der HTMLLoader-Klasse erstellen, um neue Verhalten zu erstellen. Beispielsweise können Sie eine Unterklasse erstellen, mit der Standard-Ereignis-Listener für HTMLLoader-Ereignisse definiert werden (wie etwa solche Ereignisse, die bei der Wiedergabe von HTML ausgelöst werden oder wenn der Benutzer auf einen Hyperlink klickt).

Im folgenden Beispiel wird die HTMLHost-Klasse erweitert, um *normales* Verhalten zu bieten, wenn die JavaScript-Methode `window.open()` aufgerufen wird. Anschließend wird eine Unterklasse von der HTMLLoader-Klasse definiert, die die benutzerdefinierte HTMLHost-Implementierungsklasse verwendet:

```
package
{
       import flash.html.HTMLLoader;
    public class MyHTMLHost extends HTMLHost
       {
           public function MyHTMLHost()
           {
               super(false);
        }
        override public function createWindow(opts:HTMLWindowCreateOptions):void
        {
            var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
            var bounds:Rectangle = new Rectangle(opts.x, opts.y, opts.width, opts.height);
            var html:HTMLLoader = HTMLLoader.createRootWindow(true,
                                                  initOptions,
                                                  opts.scrollBarsVisible,
                                                  bounds);
            html.stage.nativeWindow.orderToFront();
            return html
       }
}
```

Im folgenden Beispiel wird eine Unterklasse der HTMLLoader-Klasse definiert, mit der der `htmlHost`-Eigenschaft ein MyHTMLHost-Objekt zugewiesen wird:

```
package
{
       import flash.html.HTMLLoader;
    import MyHTMLHost;
    import HTMLLoader;
       public class MyHTML extends HTMLLoader
       {
           public function MyHTML()
           {
               super();
               htmlHost = new MyHTMLHost();
           }
       }
}
```

Einzelheiten zur HTMLHost-Klasse und der in diesem Beispiel verwendeten `HTMLLoader.createRootWindow()`-Methode finden Sie unter „Definieren browserähnlicher Benutzeroberflächen für HTML-Inhalt“ auf Seite 1078.

Kapitel 61: Verarbeiten HTML-bezogener Ereignisse in AIR

Adobe AIR 1.0 und höher

Ein Ereignis verarbeitendes System bietet Programmierern eine praktische Möglichkeit, auf Benutzereingaben und Systemereignisse zu reagieren. Das Adobe® AIR®-Ereignismodell ist nicht nur praktisch, sondern erfüllt auch relevante Standards. Basierend auf der Document Object Model (DOM) Level 3 Events Specification (eine Ereignis verarbeitende Architektur gemäß Branchenstandard), bietet das Ereignismodell Programmierern ein leistungsstarkes, intuitives Tool für die Ereignisverarbeitung.

HTMLLoader-Ereignisse

Adobe AIR 1.0 und höher

Ein HTMLLoader-Objekt löst die folgenden Adobe® ActionScript® 3.0-Ereignisse aus:

Ereignis	Beschreibung
`htmlDOMInitialize`	Wird ausgelöst, wenn das HTML-Dokument erstellt wird, aber bevor Skripts geparst oder DOM-Knoten zur Seite hinzugefügt werden.
`complete`	Wird ausgelöst, wenn das HTML-DOM als Reaktion auf einen Ladevorgang erstellt wurde, sofort nach dem `onload`-Ereignis in der HTML-Seite.
`htmlBoundsChanged`	Wird ausgelöst, wenn mindestens eine der Eigenschaften `contentWidth` und `contentHeight` geändert wurde.
`locationChange`	Wird ausgelöst, wenn die location-Eigenschaft des HTMLLoader geändert wurde.
`locationChanging`	Wird ausgelöst, bevor die HTMLLoader-Position sich nach einer Benutzernavigation, einem JavaScript-Aufruf oder einer Umleitung ändert. Das `locationChanging`-Ereignis wird nicht ausgelöst, wenn Sie die Methoden `load()`, `loadString()`, `reload()`, `historyGo()`, `historyForward()` oder `historyBack()` aufrufen. Bei einem Aufruf der `preventDefault()`-Methode des ausgelösten Ereignisobjekts wird die Navigation abgebrochen. Wenn ein Link im Systembrowser geöffnet wird, wird kein locationChanging-Ereignis ausgelöst, da die HTMLLoader-Position sich nicht ändert.
`scroll`	Wird ausgelöst, wenn die HTML-Engine die Bildlaufposition ändert. Scroll-Ereignisse können wegen der Navigation zu Anker-Links (#-Links) auf der Seite oder aufgrund von Aufrufen der `window.scrollTo()`-Methode auftreten. Die Eingabe von Text in ein Texteingabefeld oder einen Textbereich kann ebenfalls ein scroll-Ereignis auslösen.
`uncaughtScriptException`	Wird ausgelöst, wenn eine JavaScript-Ausnahme im HTMLLoader auftritt und die Ausnahme im JavaScript-Code nicht abgefangen wird.

Sie können auch eine ActionScript-Funktion für ein JavaScript-Ereignis (zum Beispiel `onClick`) registrieren. Ausführliche Informationen finden Sie unter „Verarbeiten von DOM-Ereignissen mit ActionScript“ auf Seite 1092.

Verarbeiten von DOM-Ereignissen mit ActionScript

Adobe AIR 1.0 und höher

Sie können ActionScript-Funktionen registrieren, um auf JavaScript-Ereignisse zu reagieren. Betrachten Sie zum Beispiel folgenden HTML-Inhalt:

```
<html>
<body>
    <a href="#" id="testLink">Click me.</a>
</html>
```

Sie können eine ActionScript-Funktion als eine Prozedur für jedes beliebige Ereignis auf der Seite registrieren. Der folgende Code fügt zum Beispiel die `clickHandler()`-Funktion als Listener für das `onclick`-Ereignis des `testLink`-Elements auf der HTML-Seite hinzu:

```
var html:HTMLLoader = new HTMLLoader( );
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.addEventListener(Event.COMPLETE, completeHandler);

function completeHandler(event:Event):void {
    html.window.document.getElementById("testLink").onclick = clickHandler;
}

function clickHandler( event:Object ):void {
    trace("Event of type: " + event.type );
}
```

Das ausgelöste Ereignisobjekt hat nicht den Typ flash.events.Event oder einer der Event-Unterklassen. Verwenden Sie die Object-Klasse, um einen Typ für das Ereignisprozedurfunktionsargument zu deklarieren.

Sie können diese Ereignisse auch mit der `addEventListener()`-Methode registrieren. Die `completeHandler()`-Methode im oben aufgeführten Beispiel lässt sich zum Beispiel durch den folgenden Code ersetzen:

```
function completeHandler(event:Event):void {
    var testLink:Object = html.window.document.getElementById("testLink");
    testLink.addEventListener("click", clickHandler);
}
```

Wenn ein Listener sich auf ein bestimmtes DOM-Element bezieht, hat es sich bewährt, zu warten, bis der übergeordnete HTMLLoader das `complete`-Ereignis auslöst, bevor die Ereignis-Listener hinzugefügt werden. HTML-Seiten laden häufig mehrere Dateien und das HTML-DOM wird nicht vollständig aufgebaut, bevor alle Dateien geladen und geparst wurden. Der HTMLLoader löst das `complete`-Ereignis aus, wenn alle Elemente erstellt wurden.

Antworten auf nicht erfasste JavaScript-Ausnahmen

Adobe AIR 1.0 und höher

Betrachten Sie den folgenden HTML-Code:

```
<html>
<head>
    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

Er enthält eine JavaScript-Funktion, `throwError()`, die auf eine unbekannte Variable verweist, `melbaToast`:

```
var x = 400 * melbaToast;
```

Wenn eine JavaScript-Operation einen unzulässigen Vorgang erkennt, der im JavaScript-Code nicht mit einer `try/catch`-Struktur erfasst wird, löst das HTMLLoader-Objekt, das die Seite enthält, ein HTMLUncaughtScriptExceptionEvent-Ereignis aus. Sie können eine Prozedur für dieses Ereignis registrieren wie im folgenden Code:

```
var html:HTMLLoader = new HTMLLoader();
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.width = container.width;
html.height = container.height;
container.addChild(html);
html.addEventListener(HTMLUncaughtScriptExceptionEvent.UNCAUGHT_SCRIPT_EXCEPTION,
                          htmlErrorHandler);
function htmlErrorHandler(event:HTMLUncaughtJavaScriptExceptionEvent):void
{
    event.preventDefault();
    trace("exceptionValue:", event.exceptionValue)
    for (var i:int = 0; i < event.stackTrace.length; i++)
    {
        trace("sourceURL:", event.stackTrace[i].sourceURL);
        trace("line:", event.stackTrace[i].line);
        trace("function:", event.stackTrace[i].functionName);
    }
}
```

In JavaScript können Sie dasselbe Ereignis mit der window.htmlLoader-Eigenschaft verarbeiten:

```
<html>
<head>
<script language="javascript" type="text/javascript" src="AIRAliases.js"></script>

    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }

        function htmlErrorHandler(event) {
            event.preventDefault();
            var message = "exceptionValue:" + event.exceptionValue + "\n";
            for (var i = 0; i < event.stackTrace.length; i++){
                message += "sourceURL:" + event.stackTrace[i].sourceURL +"\n";
                message += "line:" + event.stackTrace[i].line +"\n";
                message += "function:" + event.stackTrace[i].functionName + "\n";
            }
            alert(message);
        }

        window.htmlLoader.addEventListener("uncaughtScriptException", htmlErrorHandler);
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

Das `htmlErrorHandler()`-Ereignis bricht das Standardverhalten des Ereignisses (das im Senden der JavaScript-Fehlermeldung an die AIR-Trace-Ausgabe besteht) ab und generiert seine eigene Ausgabemeldung. Ausgegeben wird der Wert von `exceptionValue` des HTMLUncaughtScriptExceptionEvent-Objekts. Im `stackTrace`-Array werden die Eigenschaften aller Objekte ausgegeben:

```
exceptionValue: ReferenceError: Can't find variable: melbaToast
sourceURL: app:/test.html
line: 5
function: throwError
sourceURL: app:/test.html
line: 10
function: onclick
```

Verarbeiten von Laufzeitereignissen mit JavaScript

Adobe AIR 1.0 und höher

Die Laufzeitklassen unterstützen das Hinzufügen von Ereignis-Prozeduren mit der `addEventListener()`-Methode. Um eine Prozedurfunktion für ein Ereignis hinzuzufügen, rufen Sie die `addEventListener()`-Methode des Objekts auf, das das Ereignis auslöst, und geben Sie dabei den Ereignistyp und die Prozedurfunktion an. Um zum Beispiel auf das `closing`-Ereignis zu warten, das ausgelöst wird, wenn ein Benutzer auf das Schließen-Feld in der Titelleiste eines Fensters klickt, verwenden Sie die folgende Anweisung:

```
window.nativeWindow.addEventListener(air.NativeWindow.CLOSING, handleWindowClosing);
```

Erstellen einer Ereignisprozedurfunktion

Adobe AIR 1.0 und höher

Mit dem folgenden Code wird eine einfache HTML-Datei erstellt, die Informationen über die Position des Hauptfensters anzeigt. Eine Prozedurfunktion, `moveHandler()`, wartet auf ein move-Ereignis (von der NativeWindowBoundsEvent-Klasse definiert) des Hauptfensters.

```
<html>
    <script src="AIRAliases.js" />
    <script>
        function init() {
            writeValues();
            window.nativeWindow.addEventListener(air.NativeWindowBoundsEvent.MOVE,
                                                 moveHandler);
        }
        function writeValues() {
            document.getElementById("xText").value = window.nativeWindow.x;
            document.getElementById("yText").value = window.nativeWindow.y;
        }
        function moveHandler(event) {
            air.trace(event.type); // move
            writeValues();
        }
    </script>
    <body onload="init()" />
        <table>
            <tr>
                <td>Window X:</td>
                <td><textarea id="xText"></textarea></td>
            </tr>
            <tr>
                <td>Window Y:</td>
                <td><textarea id="yText"></textarea></td>
            </tr>
        </table>
    </body>
</html>
```

Wenn ein Benutzer das Fenster verschiebt, zeigen die Textfeldelemente die aktualisierten X- und Y-Positionen des Fensters an:

Beachten Sie, dass das Ereignisobjekt als Argument an die `moveHandler()`-Methode übergeben wird. Der event-Parameter ermöglicht der Prozedurfunktion, das Ereignisobjekt zu überprüfen. In diesem Beispiel verwenden Sie die `type`-Eigenschaft des Ereignisobjekts, um zu melden, dass es sich bei dem Ereignis um ein `move`-Ereignis handelt.

Entfernen von Ereignisprozeduren

Adobe AIR 1.0 und höher

Mit der `removeEventListener()`-Methode können Sie einen Ereignis-Listener entfernen, den Sie nicht mehr benötigen. Es empfiehlt sich, alle nicht mehr benötigten Listener zu entfernen. Erforderliche Parameter sind `eventName` und `listener`, die identisch sind mit den erforderlichen Parametern für die `addEventListener()`-Methode.

Entfernen von Ereignis-Listenern auf navigierenden HTML-Seiten

Adobe AIR 1.0 und höher

Wenn HTML-Inhalt Navigationsvorgänge ausführt oder wenn HTML-Inhalt verworfen wird, da das Fenster mit dem Inhalt geschlossen wird, werden die Ereignis-Listener, die auf Objekte auf der entladenen Seite verweisen, nicht automatisch entfernt. Wenn ein Objekt ein Ereignis für eine Prozedur auslöst, die bereits entladen wurde, wird die folgende Fehlermeldung angezeigt: „The application attempted to reference a JavaScript object in an HTML page that is no longer loaded.“ (Die Anwendung hat versucht, auf ein JavaScript-Objekt auf einer HTML-Seite zu verweisen, die nicht mehr geladen ist.)

Um diesen Fehler zu vermeiden, entfernen Sie JavaScript-Ereignis-Listener auf einer HTML-Seite, bevor sie entladen wird. Bei der Seitennavigation (in einem HTMLLoader-Objekt) entfernen Sie den Ereignis-Listener während des `unload`-Ereignisses des `window`-Objekts.

Mit dem folgenden JavaScript-Code wird ein Ereignis-Listener für ein `uncaughtScriptException`-Ereignis entfernt:

```
window.onunload = cleanup;
window.htmlLoader.addEventListener('uncaughtScriptException', uncaughtScriptException);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Damit der Fehler nicht beim Schließen von Fenstern mit HTML-Inhalt auftritt, rufen Sie eine Bereinigungsfunktion als Reaktion auf das `closing`-Ereignis des NativeWindow-Objekts (`window.nativeWindow`) auf. Mit dem folgenden JavaScript-Code wird ein Ereignis-Listener für ein `uncaughtScriptException`-Ereignis entfernt:

```
 window.nativeWindow.addEventListener(air.Event.CLOSING, cleanup);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Sie können diesen Fehler auch verhindern, indem Sie einen Ereignis-Listener entfernen, sobald er ausgeführt wird (falls das Ereignis nur einmal verarbeitet werden muss). Mit dem folgenden JavaScript-Code wird zum Beispiel ein Html-Fenster erstellt, indem die `createRootWindow()`-Methode der HTMLLoader-Klasse aufgerufen und ein Ereignis-Listener für das `complete`-Ereignis aufgerufen wird. Wenn die `complete`-Ereignisprozedur aufgerufen wird, entfernt sie ihren eigenen Ereignis-Listener mit der `removeEventListener()`-Funktion:

```
 var html = runtime.flash.html.HTMLLoader.createRootWindow(true);
html.addEventListener('complete', htmlCompleteListener);
function htmlCompleteListener()
{
    html.removeEventListener(complete, arguments.callee)
    // handler code..
}
html.load(new runtime.flash.net.URLRequest("second.html"));
```

Wenn Sie nicht mehr benötigte Ereignis-Listener entfernen, hat auch der Garbage Collector des Systems die Möglichkeit, den mit diesen Listenern belegten Speicher freizumachen.

Überprüfen auf vorhandene Ereignis-Listener

Adobe AIR 1.0 und höher

Mit der `hasEventListener()`-Methode können Sie prüfen, ob Ereignis-Listener für ein Objekt vorhanden sind.

Kapitel 62: Anzeigen von HTML-Inhalt in mobilen Anwendungen

Adobe AIR 2.5 und höher

Die StageWebView-Klasse zeigt HTML-Inhalt mit dem Systembrowser-Steuerelement auf Mobilgeräten und mit dem standardmäßigen Adobe® AIR® HTMLLoader-Steuerelement auf Desktopcomputern an. Überprüfen Sie anhand der `StageWebView.isSupported`-Eigenschaft, ob die Klasse auf dem jeweiligen Gerät unterstützt wird. Unterstützung ist nicht für alle Geräte im Mobilprofil garantiert.

In allen Profilen unterstützt die StageWebView-Klasse nur eingeschränkte Interaktion zwischen dem HTML-Inhalt und dem Rest der Anwendung. Sie können zwar die Navigation steuern, doch weder das Cross-Scripting noch der direkte Datenaustausch sind zulässig. Sie können Inhalt von einer lokalen URL oder einer Remote-URL laden oder einen HTML-String übergeben.

StageWebView-Objekte

Ein StageWebView-Objekt ist kein Anzeigeobjekt und kann der Anzeigeliste nicht hinzugefügt werden. Stattdessen funktioniert es als direkt an die Bühne angefügter Viewport. StageWebView-Inhalt wird über jedem Inhalt der Anzeigeliste dargestellt. Es gibt keine Möglichkeit, die Darstellungsordnung mehrerer StageWebView-Objekte zu steuern.

Zum Anzeigen eines StageWebView-Objekts weisen Sie die Bühne, auf der das Objekt angezeigt werden soll, der `stage`-Eigenschaft des StageWebView-Objekts zu. Legen Sie die Größe der Anzeige über die `viewPort`-Eigenschaft fest.

Stellen Sie die x- und y-Koordinaten der `viewPort`-Eigenschaft auf einen Wert zwischen -8192 und 8191 ein. Der maximale Wert für die Breite und Höhe der Bühne beträgt 8191. Wenn die Größe diese Höchstwerte überschreitet, wird eine Ausnahme ausgelöst.

Das folgende Beispiel erstellt ein StageWebView-Objekt, stellt die `stage`- und `viewPort`-Eigenschaften ein und zeigt einen HTML-String an:

```
var webView:StageWebView = new StageWebView();
webView.viewPort = new Rectangle( 0, 0, this.stage.stageWidth, this .stage.stageHeight);
webView.stage = this.stage;
var htmlString:String = "<!DOCTYPE HTML>" +
                        "<html><body>" +
                        "<p>King Philip could order five good steaks.</p>" +
                        "</body></html>";
webView.loadString( htmlString );
```

Um ein StageWebView-Objekt auszublenden, setzen Sie seine `stage`-Eigenschaft auf `null`. Um das Objekt vollständig zu entfernen, rufen Sie die `dispose()`-Methode auf. Der Aufruf von `dispose()` ist optional, doch wenn die Methode aufgerufen wird, kann die Speicherbereinigung den vom Objekt belegten Arbeitsspeicher schneller freigeben.

Inhalt

Inhalt kann mit zwei Methoden in ein StageWebView-Objekt geladen werden: `loadURL()` und `loadString()`.

Die `loadURL()`-Methode lädt eine Ressource, die sich an der angegebenen URL befindet. Sie können jedes URI-Schema verwenden, das vom Steuerelement für den Webbrowser des Systems unterstützt wird, wie beispielsweise data:, file:, http:, https: und javascript:. Die Schemas app: und app-storage: werden nicht unterstützt. Der URL-String wird von AIR nicht überprüft.

Die `loadString()`-Methode lädt einen Literalstring mit HTML-Inhalt. Der Speicherort einer mit dieser Methode geladenen Seite wird folgendermaßen ausgedrückt:

- Desktop: about:blank
- iOS: *htmlString*
- Android: data-URI-Format des kodierten *htmlString*

Das URI-Schema bestimmt die Regeln zum Laden von eingebetteten Inhalten oder Daten.

URI-Schema	Lokale Ressource laden	Remote-Ressource laden	XMLHttpRequest lokal	XMLHttpRequest remote
data:	Nein	Ja	Nein	Nein
file:	Ja	Ja	Ja	Ja
http:, https:	Nein	Ja	Nein	Selbe Domäne
about: (loadString()-Methode)	Nein	Ja	Nein	Nein

Hinweis: *Wenn die `displayState`-Eigenschaft der Bühne auf `FULL_SCREEN` eingestellt ist, können Sie auf einem Desktop keine Eingabe in einem Textfeld vornehmen, das im StageWebView-Objekt angezeigt wird. Dagegen können Sie unter iOS und Android auch dann Text in ein Textfeld in einem StageWebView-Objekt eingeben, wenn die `displayState`-Eigenschaft der Bühne auf `FULL_SCREEN` eingestellt ist.*

Im folgenden Beispiel wird ein StageWebView-Objekt verwendet, um die Adobe-Website anzuzeigen:

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.geom.Rectangle;

    public class StageWebViewExample extends MovieClip{

        var webView:StageWebView = new StageWebView();

        public function StageWebViewExample() {
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
            webView.loadURL( "http://www.adobe.com" );
        }
    }
}
```

Auf Android-Geräten müssen Sie die INTERNET-Berechtigung für Android angeben, damit die Anwendung Remote-Ressourcen erfolgreich laden kann.

Unter Android 3.0+ müssen Anwendungen die Hardwarebeschleunigung im manifestAdditions-Element von Android des AIR-Anwendungsdeskriptors aktivieren, damit Plugin-Inhalt in einem StageWebView-Objekt angezeigt werden kann. Einzelheiten finden Sie im Abschnitt zum Aktivieren von Flash Player und anderen Plugins in einem StageWebView-Objekt.

JavaScript-URI

Sie können eine JavaScript-URI zum Aufrufen einer Funktion verwenden, die in der HTML-Seite definiert ist, die von einem StageWebView-Objekt geladen wird. Die Funktion, die über die JavaScript-URI aufgerufen wird, wird im Kontext der geladenen Webseite ausgeführt. Im folgenden Beispiel wird ein StageWebView-Objekt zum Aufrufen einer JavaScript-Funktion verwendet:

```
package {
    import flash.display.*;
    import flash.geom.Rectangle;
    import flash.media.StageWebView;
    public class WebView extends Sprite
    {
        public var webView:StageWebView = new StageWebView();
        public function WebView()
        {
            var htmlString:String = "<!DOCTYPE HTML>" +
            "<html><script type=text/javascript>" +
            "function callURI(){" +
            "alert(\"You clicked me!!\");"+
            "}</script><body>" +
            "<p><a href=javascript:callURI()>Click Me</a></p>" +
            "</body></html>";
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
            webView.loadString( htmlString );
        }
    }
}
```

Verwandte Hilfethemen

Sean Voisen: Making the Most of StageWebView

Navigationsereignisse

Wenn der Benutzer auf einen Hyperlink in der HTML-Seite klickt, löst das StageWebView-Objekt ein `locationChanging`-Ereignis aus. Sie können die `preventDefault()`-Methode des Ereignisobjekts aufrufen, um die Navigation zu stoppen. Andernfalls navigiert das StageWebView-Objekt zu der neuen Seite und löst ein `locationChange`-Ereignis aus. Nachdem die Seite vollständig geladen wurde, löst das StageWebView-Objekt ein `complete`-Ereignis aus.

Ein `locationChanging`-Ereignis wird bei jeder HTML-Umleitung ausgelöst. Die `locationChange`- und `complete`-Ereignisse werden zum entsprechenden Zeitpunkt ausgelöst.

Unter iOS wird ein `locationChanging`-Ereignis vor einem `locationChange`-Ereignis ausgelöst, mit Ausnahme der ersten `loadURL()`- oder `loadString()`-Methode. Ein `locationChange`-Ereignis wird auch für Navigationsänderungen über Frames und Inlineframes ausgelöst.

Im folgenden Beispiel wird gezeigt, wie Sie eine Positionsänderung verhindern und die neue Seite stattdessen im Systembrowser öffnen können.

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.events.LocationChangeEvent;
    import flash.geom.Rectangle;
    import flash.net.navigateToURL;
    import flash.net.URLRequest;

    public class StageWebViewNavEvents extends MovieClip{
        var webView:StageWebView = new StageWebView();

        public function StageWebViewNavEvents() {
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
        webView.addEventListener( LocationChangeEvent.LOCATION_CHANGING, onLocationChanging );
            webView.loadURL( "http://www.adobe.com" );
        }
        private function onLocationChanging( event:LocationChangeEvent ):void
        {
            event.preventDefault();
            navigateToURL( new URLRequest( event.location ) );
        }
    }
}
```

Verlauf

Wenn der Benutzer auf Hyperlinks klickt, die im Inhalt eines StageWebView-Objekts angezeigt werden, speichert das Steuerelement den vorwärts und rückwärts gerichteten Verlaufsstapel. Das folgende Beispiel veranschaulicht die Navigation durch die beiden Verlaufsstapel. Im Beispiel werden die Softkeys für „Zurück“ und „Suchen“ verwendet.

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.geom.Rectangle;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;

    public class StageWebViewExample extends MovieClip{

        var webView:StageWebView = new StageWebView();

        public function StageWebViewExample()
        {
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
            webView.loadURL( "http://www.adobe.com" );

            stage.addEventListener( KeyboardEvent.KEY_DOWN, onKey );
        }

        private function onKey( event:KeyboardEvent ):void
        {
            if( event.keyCode == Keyboard.BACK && webView.isHistoryBackEnabled )
            {
                trace("back");
                webView.historyBack();
                event.preventDefault();
            }
            if( event.keyCode == Keyboard.SEARCH && webView.isHistoryForwardEnabled )
            {
                trace("forward");
                webView.historyForward();
            }
        }
    }
}
```

Fokus

Die StageWebView-Klasse ist zwar kein Anzeigeobjekt, sie enthält jedoch Mitglieder, die es Ihnen ermöglichen, die Fokusübergänge in das Steuerelement und aus dem Steuerelement zu verwalten.

Wenn das StageWebView-Objekt den Fokus erhält, löst es ein `focusIn`-Ereignis aus. Mit diesem Ereignis verwalten Sie bei Bedarf die Fokuselemente in Ihrer Anwendung.

Wenn das StageWebView-Objekt den Fokus wieder abgibt, löst es ein `focusOut`-Ereignis aus. Eine StageWebView-Instanz kann den Fokus abgeben, wenn ein Benutzer mit einem Trackball oder mit Pfeiltasten vor das erste oder hinter das letzte Steuerelement auf der Webseite navigiert. Über die `direction`-Eigenschaft des Ereignisobjekts können Sie feststellen, ob der Fokusablauf nach oben über den Anfang der Seite oder nach unten über das Ende der Seite hinaus geht. Anhand dieser Informationen weisen Sie den Fokus dem entsprechenden Anzeigeobjekt oberhalb oder unterhalb des StageWebView-Objekts zu.

Unter iOS kann der Fokus nicht programmgesteuert festgelegt werden. StageWebView löst `focusIn`- und `focusOut`-Ereignisse aus, wobei die direction-Eigenschaft von `FocusEvent` auf `none` eingestellt ist. Wenn der Benutzer den Bildschirm innerhalb des StageWebView-Objekts berührt, wird das `focusIn`-Ereignis ausgelöst. Berührt der Benutzer den Bildschirm außerhalb des StageWebView-Objekts, wird das `focusOut`-Ereignis ausgelöst.

Das folgende Beispiel veranschaulicht, wie der Fokus vom StageWebView-Objekt zu Flash-Anzeigeobjekten übergeht:

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.geom.Rectangle;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    import flash.text.TextField;
    import flash.text.TextFieldType;
    import flash.events.FocusEvent;
    import flash.display.FocusDirection;
    import flash.events.LocationChangeEvent;

    public class StageWebViewFocusEvents extends MovieClip{
        var webView:StageWebView = new StageWebView();
        var topControl:TextField = new TextField();
        var bottomControl:TextField = new TextField();

        public function StageWebViewFocusEvents()
        {
            trace("Starting");
            topControl.type = TextFieldType.INPUT;
            addChild( topControl );
            topControl.height = 60;
            topControl.width = stage.stageWidth;
            topControl.background = true;
            topControl.text = "One control on top.";
            topControl.addEventListener( FocusEvent.FOCUS_IN, flashFocusIn );
            topControl.addEventListener( FocusEvent.FOCUS_OUT, flashFocusOut );

            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 60, stage.stageWidth, stage.stageHeight
- 120 );
            webView.addEventListener( FocusEvent.FOCUS_IN, webFocusIn );
            webView.addEventListener(FocusEvent.FOCUS_OUT, webFocusOut );
            webView.addEventListener(LocationChangeEvent.LOCATION_CHANGING,
                                     function( event:LocationChangeEvent ):void
                                     {
                                        event.preventDefault();
                                     } );
            webView.loadString("<form action='#'><input/><input/><input/></form>");
            webView.assignFocus();

            bottomControl.type = TextFieldType.INPUT;
            addChild( bottomControl );
            bottomControl.y = stage.stageHeight - 60;
            bottomControl.height = 60;
            bottomControl.width = stage.stageWidth;
            bottomControl.background = true;
            bottomControl.text = "One control on the bottom.";
            bottomControl.addEventListener( FocusEvent.FOCUS_IN, flashFocusIn );
```

```
            bottomControl.addEventListener( FocusEvent.FOCUS_OUT, flashFocusOut );}

        private function webFocusIn( event:FocusEvent ):void
        {
            trace("Web focus in");
        }

        private function webFocusOut( event:FocusEvent ):void
        {
            trace("Web focus out: " + event.direction);
            if( event.direction == FocusDirection.TOP )
            {
                stage.focus = topControl;
            }
            else
            {
                stage.focus = bottomControl;
            }
        }

        private function flashFocusIn( event:FocusEvent ):void
        {
            trace("Flash focus in");
            var textfield:TextField = event.target as TextField;
            textfield.backgroundColor = 0xff5566;
        }

        private function flashFocusOut( event:FocusEvent ):void
        {
            trace("Flash focus out");
            var textfield:TextField = event.target as TextField;
            textfield.backgroundColor = 0xffffff;
        }

    }
}
```

Bitmaperfassung

Ein StageWebView-Objekt wird über dem gesamten Inhalt der Anzeigeliste gerendert. Es ist nicht möglich, oberhalb eines StageWebView-Objekts weiteren Inhalt hinzuzufügen. Beispielsweise kann über dem StageWebView-Inhalt kein Dropdownmenü eingeblendet werden. Um dieses Problem zu beheben, erfassen Sie einen Schnappschuss des StageWebView-Objekts. Dann blenden Sie das StageWebView-Objekt aus und fügen stattdessen den Bitmap-Schnappschuss hinzu.

Das folgende Beispiel zeigt, wie Sie einen Schnappschuss eines StageWebView-Objekts mithilfe der `drawViewPortToBitmapData`-Methode erfassen. Das StageWebView-Objekt wird ausgeblendet, indem die Bühne auf „null“ gesetzt wird. Nachdem die Webseite vollständig geladen wurde, wird eine Funktion aufgerufen, die die Bitmap erfasst und anzeigt. Bei der Ausführung zeigt der Code zwei Beschriftungen an: Google und Facebook. Durch Klicken auf eine Beschriftung wird die entsprechende Webseite erfasst und als Schnappschuss auf der Bühne angezeigt.

```
package
{
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.geom.Rectangle;
    import flash.media.StageWebView;
    import flash.net.*;
    import flash.text.TextField;
    public class stagewebview extends Sprite
    {
        public var webView:StageWebView=new StageWebView();
        public var textGoogle:TextField=new TextField();
        public var textFacebook:TextField=new TextField();
        public function stagewebview()
        {
            textGoogle.htmlText="<b>Google</b>";
            textGoogle.x=300;
            textGoogle.y=-80;
            addChild(textGoogle);
            textFacebook.htmlText="<b>Facebook</b>";
            textFacebook.x=0;
            textFacebook.y=-80;
            addChild(textFacebook);
            textGoogle.addEventListener(MouseEvent.CLICK,goGoogle);
            textFacebook.addEventListener(MouseEvent.CLICK,goFaceBook);
            webView.stage = this.stage;
            webView.viewPort = new Rectangle(0, 0, stage.stageWidth, stage.stageHeight);
        }
    public function goGoogle(e:Event):void
        {
            webView.loadURL("http://www.google.com");
            webView.stage = null;
            webView.addEventListener(Event.COMPLETE,handleLoad);
        }

    public function goFaceBook(e:Event):void
        {
            webView.loadURL("http://www.facebook.com");
            webView.stage = null;
            webView.addEventListener(Event.COMPLETE,handleLoad);
        }
    public function handleLoad(e:Event):void
        {
            var bitmapData:BitmapData = new BitmapData(webView.viewPort.width,
webView.viewPort.height);
            webView.drawViewPortToBitmapData(bitmapData);
            var webViewBitmap:Bitmap=new Bitmap(bitmapData);
            addChild(webViewBitmap);
        }
    }
}
```

Kapitel 63: Verwenden von Workern für Parallelität

Flash Player 11.4 und höher, Adobe AIR 13.4 und höher für Desktopplattformen

ActionScript-Worker ermöglichen die gleichzeitige (parallele) Ausführung von Code. Anders ausgedrückt kann Code im Hintergrund ausgeführt werden, ohne dass die Ausführung des Hauptcodes unterbrochen wird.

Die ActionScript-APIs für die Parallelität sind nur auf Desktopplattformen in Flash Player 11.4 und höher sowie AIR 3.4 und höher verfügbar. Parallelität wird nicht in AIR für mobile Plattformen unterstützt.

Informationen zu Workern und Parallelität

Flash Player 11.4 und höher, Adobe AIR 13.4 und höher für Desktopplattformen

Bei Anwendungen, die keine Worker verwenden, wird der Anwendungscode in einem einzelnen, linearen Block von Ausführungsschritten ausgeführt, der als Ausführungs*thread* bezeichnet wird. Der Thread führt den Code aus, den ein Entwickler geschrieben hat. Außerdem führt er viel von dem Code aus, der zur Laufzeitumgebung gehört, vor allem den Code, der den Bildschirm aktualisiert, wenn sich die Eigenschaften von Anzeigeobjekten ändern. Zwar wird Code in größeren Abschnitten wie Methoden und Klassen geschrieben, zur Laufzeit wird der Code jedoch Zeile für Zeile ausgeführt, als ob er in einer einzelnen, langen Folge von Schritten geschrieben worden wäre. Hypothetisches Beispiel der Schritte, die von einer Anwendung ausgeführt werden:

1. Frameeintritt: Die Laufzeitumgebung ruft alle vorhandenen `enterFrame`-Ereignisprozeduren auf und führt ihren Code nacheinander aus
2. Mausereignis: Der Benutzer bewegt die Maus und die Laufzeitumgebung ruft alle Mausereignisprozeduren auf, während die verschiedenen Rollover- und Rollout-Ereignisse auftreten
3. Ereignis „Ladevorgang abgeschlossen“: Eine Anforderung, eine XML-Datei aus einer URL zu laden, gibt die geladenen Dateidaten zurück. Die Ereignisprozedur wird aufgerufen und führt ihre Schritte aus. Dabei wird der XML-Inhalt gelesen und es werden Objekte aus den XML-Daten erstellt.
4. Mausereignis: Die Maus wurde erneut bewegt, sodass die Laufzeitumgebung die entsprechenden Ereignisprozeduren aufruft
5. Rendering: Es warten keine weiteren Ereignisse, sodass die Laufzeitumgebung den Bildschirm entsprechend den Änderungen an Anzeigeobjekten aktualisiert
6. Frameeintritt: Der Zyklus beginnt erneut

Wie im Beispiel beschrieben, werden die hypothetischen Schritte 1-5 nacheinander innerhalb eines einzelnen Zeitblocks (Frame genannt) ausgeführt. Da sie nacheinander in einem einzelnen Thread ausgeführt werden, kann die Laufzeitumgebung nicht einen Schritt des Prozesses unterbrechen, um einen anderen auszuführen. Bei einer Framerate von 30 fps hat die Laufzeitumgebung weniger als eine Dreißigstelsekunde Zeit, um alle diese Operationen auszuführen. Häufig reicht dies zum Ausführen des Codes, und die Laufzeitumgebung wartet einfach während der

verbleibenden Zeit. Stellen Sie sich jedoch die XML-Daten, die in Schritt 3 geladen werden, einmal als sehr große, sehr verschachtelte XML-Struktur vor. Es kann merklich länger als eine Dreißigstelsekunde dauern, bis der Code die XML durchläuft und Objekte erstellt. In diesem Fall passieren die späteren Schritte (Antwort auf die Mausaktion und Neuzeichnen des Bildschirms) nicht so schnell wie gewünscht. Dadurch friert der Bildschirm ein und kommt ins Stocken, da der Bildschirm nicht schnell genug neu gezeichnet wird, wenn der Benutzer die Maus bewegt.

Wenn der gesamte Code in demselben Thread ausgeführt wird, gibt es nur eine Möglichkeit, gelegentliches Stocken und Einfrieren zu vermeiden: Es dürfen keine Operationen, die sehr lange dauern, ausgeführt werden, zum Beispiel Wiederholschleifen durch eine große Datenmenge. ActionScript-Worker bieten eine andere Lösung an. Code mit langer Ausführungsdauer kann in einem separaten Worker ausgeführt werden. Jeder Worker wird in einem separaten Thread ausgeführt, sodass der Hintergrundworker den Code mit langer Ausführungsdauer in einem eigenen Thread ausführt. So kann im Ausführungsthread des Hauptworkers der Bildschirm neu gezeichnet werden, ohne dass der Vorgang durch andere Operationen blockiert wird.

Die Möglichkeit, auf diese Weise mehrere Codeoperationen zur gleichen Zeit auszuführen, wird als *Parallelität* bezeichnet. Wenn der Hintergrundworker seine Operation abgeschlossen hat, können dem Hauptworker Benachrichtigungen und Daten gesendet werden. Dies kann auch bei bestimmten „Fortschrittspunkten“ während der Operation erfolgen. So können Sie Code schreiben, der komplexe oder zeitaufwändige Operationen ausführt, ohne dass der Benutzer erleben muss, dass der Bildschirm einfriert.

Worker sind hilfreich, da sie die Wahrscheinlichkeit verringern, dass die Framerate aufgrund der Blockierung des Hauptrenderingthreads durch anderen Code abfällt. Auf der anderen Seite erfordern Worker zusätzlichen Systemarbeitsspeicher und CPU-Rechenleistung, was die allgemeine Anwendungsleistung verringern kann. Da jeder Worker seine eigene Instanz der virtuellen Maschine der Laufzeitumgebung verwendet, kann der Mehraufwand auch bei trivialen Workern erheblich ins Gewicht fallen. Wenn Sie Worker verwenden, testen Sie Ihren Code auf allen Zielplattformen, um sicherzustellen, dass die Anforderungen an das System nicht zu groß sind. Adobe empfiehlt, in einem typischen Szenario nicht mehr als ein oder zwei Hintergrundworker zu verwenden.

Erstellen und Verwalten von Workern

Flash Player 11.4 und höher, Adobe AIR 13.4 und höher für Desktopplattformen

Der erste Schritt bei der Verwendung eines Workers für die Parallelität besteht darin, einen Hintergrundworker zu erstellen. Worker lassen sich mit zwei Arten von Objekten erstellen. Das erste ist eine Worker-Instanz, die Sie erstellen. Das andere ist ein WorkerDomain-Objekt, welches den Worker erstellt und die ausgeführten Worker-Objekte in einer Anwendung verwaltet.

Wenn die Laufzeitumgebung geladen wird, erstellt sie das WorkerDomain-Objekt automatisch. Die Laufzeitumgebung erstellt auch automatisch einen Worker für die Haupt-SWF-Datei der Anwendung. Dieser erste Worker wird *ursprünglicher Worker* genannt.

Da es für eine Anwendung nur ein WorkerDomain-Objekt gibt, greifen Sie mit der statischen `WorkerDomain.current`-Eigenschaft auf die WorkerDomain-Instanz zu.

Sie können jederzeit auf die aktuelle Worker-Instanz (der Worker, in dem der aktuelle Code ausgeführt wird) zugreifen, indem Sie die statische `Worker.current`-Eigenschaft verwenden.

Erstellen eines Worker-Objekts aus einer SWF-Datei

So wie die Haupt-SWF-Datei innerhalb des ursprünglichen Workers ausgeführt wird, führt ein Hintergrundworker den Code einer einzelnen SWF-Datei aus. Um einen Hintergrundworker zu verwenden, müssen Sie den Code des Workers als SWF-Datei schreiben und kompilieren. Damit der Hintergrundworker erstellt werden kann, benötigt der übergeordnete Worker Zugriff auf die Bytes dieser SWF-Datei, und zwar in Form eines ByteArray-Objekts. Sie übergeben dieses ByteArray an die `createWorker()`-Methode des WorkerDomain-Objekts, um den Worker zu erstellen.

Es gibt im Wesentlichen drei Möglichkeiten, die SWF-Datei des Hintergrundworkers als ByteArray-Objekt zu erhalten:

Einbetten der Worker-SWF-Datei

Verwenden Sie das [Embed]-Metatag, um die Worker-SWF-Datei als ByteArray in die Haupt-SWF-Datei einzubetten:

```
[Embed(source="../swfs/BgWorker.swf", mimeType="application/octet-stream")]
private static var BgWorker_ByteClass:Class;
private function createWorker():void
{
    var workerBytes:ByteArray = new BgWorker_ByteClass();
    var bgWorker:Worker = WorkerDomain.current.createWorker(workerBytes);

    // ... set up worker communication and start the worker
}
```

Die Worker-SWF-Datei wird als ByteArray-Unterklasse mit dem Namen BgWorker_ByteClass kompiliert. Indem Sie eine Instanz dieser Klasse erstellen, erhalten Sie ein ByteArray, das mit den Bytes der Worker-SWF-Datei gefüllt ist.

Laden einer externen Worker-SWF-Datei

Verwenden Sie ein URLLoader-Objekt, um eine externe SWF-Datei zu laden. Die SWF-Datei muss aus derselben Sicherheitsdomäne stammen, zum Beispiel eine SWF-Datei, die aus derselben Internetdomäne wie die Haupt-SWF-Datei geladen wird oder die in einem AIR-Anwendungspaket enthalten ist.

```
var workerLoader:URLLoader = new URLLoader();
workerLoader.dataFormat = URLLoaderDataFormat.BINARY;
workerLoader.addEventListener(Event.COMPLETE, loadComplete);
workerLoader.load(new URLRequest("BgWorker.swf"));

private function loadComplete(event:Event):void
{
    // create the background worker
    var workerBytes:ByteArray = event.target.data as ByteArray;
    var bgWorker:Worker = WorkerDomain.current.createWorker(workerBytes);

    // ... set up worker communication and start the worker
}
```

Wenn der URLLoader mit dem Laden der SWF-Datei fertig ist, sind die Bytes der SWF-Datei in der `data`-Eigenschaft des URLLoader-Objekts verfügbar (in diesem Beispiel `event.target.data`).

Verwenden der Haupt-SWF-Datei als Worker-SWF-Datei

Sie können eine einzelne SWF-Datei sowohl als Haupt-SWF-Datei als auch als Worker-SWF-Datei verwenden. Verwenden Sie die `loaderInfo.bytes`-Eigenschaft der Hauptanzeigeklasse, um auf die Bytes der SWF-Datei zuzugreifen.

```
// The primordial worker's main class constructor
public function PrimordialWorkerClass()
{
    init();
}

private function init():void
{
    var swfBytes:ByteArray = this.loaderInfo.bytes;

    // Check to see if this is the primordial worker or the background worker
    if (Worker.current.isPrimordial)
    {
        // create a background worker
        var bgWorker:Worker = WorkerDomain.current.createWorker(swfBytes);

        // ... set up worker communication and start the worker
    }
    else // entry point for the background worker
    {
        // set up communication between workers using getSharedProperty()
        // ... (not shown)

        // start the background work
    }
}
```

Wenn Sie diese Technik einsetzen, verwenden Sie eine `if`-Anweisung, um den Code der SWF-Datei innerhalb des Konstruktors der Hauptklasse oder einer von dieser aufgerufenen Methode zu verzweigen. Um festzustellen, ob der Code im Hauptworker oder im Hintergrundworker ausgeführt wird, überprüfen Sie die `isPrimordial`-Eigenschaft des aktuellen Worker-Objekts wie im Beispiel gezeigt.

Starten der Ausführung eines Workers

Nachdem Sie einen Worker erstellt haben, starten Sie die Ausführung seines Codes, indem Sie die `start()`-Methode des Workers aufrufen. Die `start()`-Operation beginnt nicht sofort. Um festzustellen, wann der Worker ausgeführt wird, registrieren Sie einen Listener für das `workerState`-Ereignis des Worker-Objekts. Dieses Ereignis wird abgesetzt, wenn das Worker-Objekt den Zustand in seinem Lebenszyklus ändert, zum Beispiel wenn mit dem Ausführen des Codes begonnen wird. Überprüfen Sie in Ihrer `workerState`-Ereignisprozedur, dass die `state`-Eigenschaft des Worker-Objekts den Wert `WorkerState.RUNNING` aufweist. An diesem Punkt wird der Worker ausgeführt, und sein Hauptklassenkonstruktor wurde ausgeführt. Das folgende Codebeispiel zeigt das Registrieren eines Listeners für das `workerState`-Ereignis und das Aufrufen der `start()`-Methode:

```
// listen for worker state changes to know when the worker is running
bgWorker.addEventListener(Event.WORKER_STATE, workerStateHandler);
// set up communication between workers using
// setSharedProperty(), createMessageChannel(), etc.
// ... (not shown)
bgWorker.start();
private function workerStateHandler(event:Event):void
{
    if (bgWorker.state == WorkerState.RUNNING)
    {
        // The worker is running.
        // Send it a message or wait for a response.
    }
}
```

Verwalten der Workerausführung

Sie können jederzeit auf die Gruppe der ausgeführten Worker in Ihrer Anwendung zugreifen, indem Sie die `listWorkers()`-Methode der WorkerDomain-Methode verwenden. Diese Methode gibt die Gruppe der Worker zurück, deren `state`-Eigenschaft den Wert `WorkerState.RUNNING` aufweist, dazu gehört auch der ursprüngliche Worker. Wenn ein Worker nicht gestartet oder seine Ausführung bereits beendet wurde, wird er nicht zurückgegeben.

Wenn Sie einen Worker nicht mehr benötigen, können Sie die `terminate()`-Methode des Worker-Objekts aufrufen, um den Worker zu schließen und seinen Speicher sowie andere Systemressourcen freizugeben.

Kommunikation zwischen Workern

Flash Player 11.4 und höher, Adobe AIR 13.4 und höher für Desktopplattformen

Zwar führen Worker ihren Code in separaten Ausführungsthreads aus, dies wäre aber wenig hilfreich, wenn sie vollständig voneinander isoliert wären. Kommunikation zwischen Workern bedeutet letztendlich das Übergeben von Daten zwischen Workern. Es gibt drei Mechanismen, um Daten von einem Worker zu einem anderen zu bringen.

Bei der Überlegung, welche dieser Techniken zur gemeinsamen Datennutzung für eine bestimmte Datenübergabe die richtige ist, ziehen Sie in Betracht, worin sie sich unterscheiden. Ein Unterschied ist, ob es ein Ereignis gibt, um den Empfänger zu informieren, wenn neue Daten vorliegen, oder ob der empfangende Worker auf Updates überprüfen muss. Ein anderer Unterschied besteht darin, wie die Daten tatsächlich übergeben werden. In einigen Fällen erhält der empfangende Worker eine Kopie der gemeinsam genutzten Daten, was bedeutet, dass mehr Objekte erstellt werden und somit Arbeitsspeicher und CPU stärker beansprucht werden. In anderen Fällen greifen die Worker auf Objekte zu, die auf denselben zugrunde liegenden Systemspeicher verweisen, sodass weniger Objekte erstellt werden und insgesamt weniger Speicher benötigt wird. Diese Unterschiede sind hier herausgestellt:

Kommunikationstechnik	Setzt Ereignis ab, wenn Daten empfangen werden	Nutzt Arbeitsspeicher gemeinsam mit Workern
Von Workern geteilte Eigenschaften	Nein	Nein, Objekte sind Kopien, keine Verweise
MessageChannel	Ja	Nein, Objekte sind Kopien, keine Verweise
Gemeinsam nutzbares ByteArray	Nein	Ja, Speicher wird gemeinsam genutzt

Übergeben von Daten mit einer gemeinsam genutzten Eigenschaft

Die einfachste Möglichkeit, Daten zwischen Workern zu teilen, ist die Verwendung einer gemeinsam genutzten Eigenschaft. Jeder Worker pflegt ein internes Wörterbuch von Werten gemeinsam genutzter Eigenschaften. Diese Eigenschaften werden mit String-Schlüsselnamen gespeichert, um sie zu unterscheiden. Um ein Objekt bei einem Worker als gemeinsam genutzte Eigenschaft zu speichern, rufen Sie die `setSharedProperty()`-Methode des Worker-Objekts mit zwei Argumenten auf, nämlich dem Schlüsselnamen und dem zu speichernden Wert:

```
// code running in the parent worker
bgWorker.setSharedProperty("sharedPropertyName", someObject);
```

Nachdem die gemeinsam genutzte Eigenschaft eingerichtet wurde, kann der Wert durch einen Aufruf der `getSharedProperty()`-Methode des Worker-Objekts gelesen werden, wobei der Schlüsselname übergeben wird:

```
// code running in the background worker
receivedProperty = Worker.current.getSharedProperty("sharedPropertyName");
```

Es gibt keine Einschränkungen dafür, welcher Worker den Wert der Eigenschaft liest oder festlegt. Zum Beispiel kann Code in einem Hintergrundworker seine `setSharedProperty()`-Methode aufrufen, um einen Wert zu speichern. Code, der im übergeordneten Worker ausgeführt wird, kann dann die `getSharedProperty()`-Eigenschaft verwenden, um die Daten abzurufen.

Der Wert, der an die `setSharedProperty()`-Methode übergeben wird, kann nahezu jeder Objekttyp sein. Wenn Sie die `getSharedProperty()`-Methode aufrufen, ist das Objekt, das zurückgegeben wird, eine Kopie des Objekts, das in `setSharedProperty()` übergeben wurde, und nur in einigen wenigen Sonderfällen ein Verweis auf dasselbe Objekt. Einzelheiten zum Datenaustausch werden unter „Gemeinsam genutzte Verweise und kopierte Werte“ auf Seite 1113 beschrieben.

Der größte Vorteil beim Verwenden einer gemeinsam genutzten Eigenschaft zum Übergeben von Daten zwischen Workern besteht darin, dass diese Methode schon verfügbar ist, bevor der Worker ausgeführt wird. Sie können die `setSharedProperty()`-Methode eines Hintergrundworkerobjekts aufrufen, um eine gemeinsam genutzte Eigenschaft festzulegen, bevor der Worker ausgeführt wird. Wenn der übergeordnete Worker die `start()`-Methode des Worker-Objekts aufruft, ruft die Laufzeitumgebung den Hauptklassenkonstruktor des untergeordneten Workers auf. Alle gemeinsam genutzten Eigenschaften, die vor dem Aufruf von `start()` festgelegt wurden, können vom Code im untergeordneten Worker gelesen werden.

Übergeben von Daten über einen MessageChannel

Ein Nachrichtenkanal ist eine unidirektionale Verbindung zur Datenübergabe zwischen zwei Workern. Das Verwenden eines MessageChannel-Objekts zum Übergeben von Daten zwischen Workern hat einen wichtigen Vorteil. Wenn Sie eine Nachricht (ein Objekt) über einen Nachrichtenkanal senden, setzt das MessageChannel-Objekt ein `channelMessage`-Ereignis ab. Code im empfangenden Worker kann einen Listener für dieses Ereignis verwenden, um festzustellen, wann Daten verfügbar sind. Auf diese Weise muss der empfangende Worker nicht ständig prüfen, ob Datenaktualisierungen vorliegen.

Ein Nachrichtenkanal ist nur mit zwei Workern verknüpft, einem Sender und einem Empfänger. Um ein MessageChannel-Objekt zu erstellen, rufen Sie die `createMessageChannel()`-Methode des sendenden Worker-Objekts auf und übergeben den empfangenden Worker als Argument:

```
// In the sending worker swf
var sendChannel:MessageChannel;
sendChannel = Worker.current.createMessageChannel(receivingWorker);
```

Beide Worker benötigen Zugriff auf das MessageChannel-Objekt. Die einfachste Möglichkeit, dafür zu sorgen, ist die Übergabe des MessageChannel-Objekts mit der `setSharedProperty()`-Methode:

```
receivingWorker.setSharedProperty("incomingChannel", sendChannel);
```

Registrieren Sie im empfangenden Worker einen Listener für das `channelMessage`-Ereignis des MessageChannel-Objekts. Dieses Ereignis wird abgesetzt, wenn der sendende Worker Daten über den Nachrichtenkanal sendet.

```
// In the receiving worker swf
var incomingChannel:MessageChannel;
incomingChannel = Worker.current.getSharedProperty("incomingChannel");
incomingChannel.addEventListener(Event.CHANNEL_MESSAGE, handleIncomingMessage);
```

Zum eigentlichen Senden der Daten rufen Sie im sendenden Worker die `send()`-Methode des MessageChannel-Objekts auf:

```
// In the sending worker swf
sendChannel.send("This is a message");
```

Im empfangenden Worker ruft das MessageChannel-Objekt die `channelMessage`-Ereignisprozedur auf. Der empfangende Worker kann die Daten dann abrufen, indem er die `receive()`-Methode des MessageChannel-Objekts aufruft.

```
private function handleIncomingMessage(event:Event):void
{
    var message:String = incomingChannel.receive() as String;
}
```

Das von der receive()-Methode zurückgegebene Objekt weist denselben Datentyp auf wie das Objekt, das an die `send()`-Methode übergeben wurde. Das empfangene Objekt ist eine Kopie des Objekts, das vom Sender übergeben wurde und kein Verweis auf das Objekt im sendenden Worker. Eine Ausnahme sind lediglich einige wenige Datentypen, die unter „Gemeinsam genutzte Verweise und kopierte Werte“ auf Seite 1113 beschrieben werden.

Teilen von Daten mit einem gemeinsam nutzbaren ByteArray

Wenn ein Objekt zwischen zwei Workern übergeben wird, erhält der empfangende Worker ein neues Objekt, das eine Kopie des Originalobjekts ist. Die beiden Objekte werden an unterschiedlichen Orten im Systemspeicher gespeichert. Jede Kopie des empfangenen Objekts erhöht demzufolge die Gesamtspeichernutzung durch die Laufzeitumgebung. Außerdem betreffen Änderungen, die Sie in einem Worker an einem Objekt vornehmen, die Kopie dieses Objekts im anderen Worker nicht. Ausführlichere Informationen dazu, wie Daten kopiert werden, finden Sie unter „Gemeinsam genutzte Verweise und kopierte Werte“ auf Seite 1113.

Standardmäßig verwendet ein ByteArray-Objekt dasselbe verhalten. Wenn Sie eine ByteArray-Instanz an die `setSharedProperty()`-Methode eines Worker-Objekts oder an die `send()`-Methode eines MessageChannel-Objekts übergeben, erstellt die Laufzeitumgebung ein neues ByteArray im Arbeitsspeicher des Computers. Der empfangende Worker erhält eine ByteArray-Instanz, die ein Verweis auf dieses neue ByteArray ist. Sie können dieses Verhalten eines ByteArray-Objekts ändern, indem Sie seine `shareable`-Eigenschaft auf `true` einstellen.

Wenn ein gemeinsam nutzbares ByteArray-Objekt von einem Worker zum anderen übergeben wird, ist die ByteArray-Instanz im empfangenden Worker ein Verweis auf denselben zugrunde liegenden Systemspeicher, der von der ByteArray-Instanz im sendenden Worker verwendet wird. Wenn Code in einem Worker den Inhalt des Bytearrays ändert, sind diese Änderungen sofort in anderen Workern verfügbar, die Zugriff auf dieses gemeinsam genutzte Bytearray haben.

Da Worker ihren Code gleichzeitig ausführen, ist es möglich, dass zwei Worker zur selben Zeit versuchen, auf dieselben Bytes in einem Bytearray zuzugreifen. Dies kann zu Datenverlust oder Beschädigungen führen. Es gibt verschiedene APIs, mit denen Sie den Zugriff auf gemeinsam genutzte Ressourcen verwalten und derartige Probleme vermeiden können.

Die ByteArray-Klasse enthält Methoden, mit denen Sie den Inhalt des Bytearrays in einem einzigen Vorgang validieren und ändern können:

- atomicCompareAndSwapIntAt()-Methode
- atomicCompareAndSwapLength()-Methode

Zusätzlich enthält das flash.concurrent-Paket Klassen, die die Zugriffssteuerung beim Arbeiten mit gemeinsam genutzten Ressourcen ermöglichen:

- Mutex-Klasse
- Condition-Klasse

Gemeinsam genutzte Verweise und kopierte Werte

Wenn Sie `Worker.setSharedProperty()` oder `MessageChannel.send()` aufrufen, wird das Objekt, das an den empfangenden Worker übergeben wird, im Normalfall übergeben, indem es im AMF-Format serialisiert wird. Dies hat einige Konsequenzen:

- Das Objekt, das im empfangenden Worker erstellt wird, wenn die `getSharedProperty()`-Methode aufgerufen wird, wird aus den AMF-Bytes deserialisiert. Es handelt sich um eine Kopie des Originalobjekts, nicht um einen Verweis auf das Objekt. Änderungen, die in einem der beiden Worker am Objekt vorgenommen werden, erfolgen nicht in der Kopie im jeweils anderen Worker.
- Objekte, die nicht in das AMF-Format serialisiert werden können, wie zum Beispiel Anzeigeobjekte, können nicht mit den Methoden `Worker.setSharedProperty()` oder `MessageChannel.send()` an einen Worker übergeben werden.
- Damit eine benutzerdefinierte Klasse korrekt deserialisiert werden kann, muss die Klassendefinition mit der `flash.net.registerClassAlias()`-Funktion oder unter Verwendung von `[RemoteClass]`-Metadaten registriert werden. Es muss derselbe Alias für die Klassenversionen beider Worker verwendet werden.

Es gibt fünf Sonderfälle von Objekten, die tatsächlich gemeinsam genutzt und nicht zwischen Workern kopiert werden:

- Worker-Objekte
- MessageChannel-Objekte
- gemeinsam nutzbares Bytearray (ein ByteArray-Objekt, dessen `shareable`-Eigenschaft den Wert `true` hat)
- Mutex-Objekte
- Condition-Objekte

Wenn Sie eine Instanz eines dieser Objekte mit der `Worker.setSharedProperty()`-Methode oder der `MessageChannel.send()`-Methode übergeben, hat jeder Worker einen Verweis auf dasselbe zugrunde liegende Objekt. Änderungen, die in einem Worker an einer Instanz vorgenommen werden, sind sofort auch in den anderen Workern verfügbar. Wenn Sie dieselbe Instanz eines dieser Objekte mehrmals an einen Worker übergeben, erstellt die Laufzeitumgebung keine neue Kopie des Objekts im empfangenden Worker. Stattdessen wird derselbe Verweis wiederverwendet.

Weitere Techniken der gemeinsamen Datennutzung

Neben den workerspezifischen Mechanismen für die Datenübergabe können Worker Daten auch mit allen vorhanden APIs austauschen, die die gemeinsame Datennutzung zwischen zwei SWF-Dateien unterstützen, zum Beispiel:

- lokale gemeinsame Objekte

- Daten in eine Datei in einem Worker schreiben und aus der Datei in einem anderen Worker lesen
- Daten in eine SQLite-Datenbank schreiben und Daten daraus lesen

Wenn Sie eine Ressource mit mehreren Workern gemeinsam nutzen, müssen Sie im Allgemeinen vermeiden, dass mehrere Worker zur selben Zeit auf die Ressource zugreifen. Es kann zum Beispiel zu Datenverlusten oder -beschädigungen kommen, wenn mehrere Worker gleichzeitig auf eine Datei im lokalen Dateisystem zugreifen. Dies wird auch nicht von allen Betriebssystemen unterstützt.

Zur Vermeidung von Problemen mit gleichzeitigem (parallelem) Zugriff verwenden Sie die Mutex- und die Condition-Klasse im flash.concurrent-Paket, um den Zugriff auf gemeinsam genutzte Ressourcen zu steuern.

Anders als andere Mechanismen für die gemeinsame Datennutzung ist die SQLite-Datenbank für den gleichzeitigen (parallelen) Zugriff ausgelegt und verfügt über eigene integrierte Transaktionsunterstützung. Mehrere Worker können auf eine SQLite-Datenbank zugreifen, ohne die Beschädigung der Daten zu riskieren. Da die Worker unterschiedliche SQLConnection-Instanzen verwenden, greift jeder Worker in einer separaten Transaktion auf die Datenbank zu. Die Integrität der Daten wird durch die gleichzeitig stattfindenden Datenbearbeitungsoperationen nicht beeinträchtigt.

Siehe auch

„Arbeiten mit lokalen SQL-Datenbanken in AIR“ auf Seite 761
flash.concurrent-Paket

Kapitel 64: Sicherheit

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sicherheit ist für Adobe, Benutzer, die Eigentümer von Websites und Entwickler von Inhalten von entscheidender Bedeutung. Aus diesem Grund umfassen Adobe® Flash® Player und Adobe® AIR™ einen umfangreichen Satz an Sicherheitsrichtlinien und -steuerungen, um Benutzer, die Eigentümer von Websites und Entwickler von Inhalten zu schützen. Dieser Abschnitt behandelt das Sicherheitsmodell für SWF-Dateien, die mit ActionScript 3.0 veröffentlicht und in Flash Player 9.0.124.0 oder höher ausgeführt werden, sowie für SWF-, HTML- und JavaScript-Dateien, die in AIR 1.0 oder höher ausgeführt werden, sofern nicht anders angegeben.

Diesem Abschnitt bietet einen Überblick über die Sicherheit, aber es werden nicht alle Implementierungsdetails, Nutzungsszenarien oder Auswirkungen der Verwendung bestimmter APIs ausführlich beschrieben. Eine detaillierte Beschreibung der Flash Player-Sicherheitskonzepte finden Sie im Abschnitt „Sicherheit“ des Flash Player Developer Center unter www.adobe.com/go/devnet_security_de.

Überblick über die Sicherheit der Flash-Plattform

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Großteil des Sicherheitsmodells, das von Flash Player und AIR-Laufzeitumgebungen verwendet wird, basiert auf der Ursprungsdomäne für geladene SWF-Dateien, HTML-Dateien, Medien und andere Elemente. Ausführbarer Code in einer Datei aus einer bestimmten Internetdomäne, beispielsweise www.example.com, kann immer auf alle Daten aus dieser Domäne zugreifen. Diese Daten werden in der gleichen Sicherheitsgruppe abgelegt, die auch als *Sicherheits-Sandbox* bezeichnet wird. (Weitere Informationen finden Sie unter „Sicherheits-Sandboxen“ auf Seite 1117.)

Beispielsweise kann ActionScript-Code in einer SWF-Datei andere SWF-Dateien, Bitmap-, Audio- und Textdateien sowie andere Daten aus der eigenen Domäne laden. Darüber hinaus ist jederzeit ein Cross-Scripting zwischen SWF-Dateien aus der gleichen Domäne zulässig, sofern beide Dateien mit ActionScript 3.0 geschrieben wurden. Beim *Cross-Scripting* kann der Code in einer Datei auf die Eigenschaften, Methoden und Objekte zugreifen, die vom Code in einer anderen Datei definiert sind.

Cross-Scripting wird jedoch nicht zwischen SWF-Dateien unterstützt, von denen eine mit ActionScript 3.0 und die andere mit früheren Versionen von ActionScript geschrieben wurde. Diese Dateien können jedoch über die LocalConnection-Klasse miteinander kommunizieren. Weiterhin ist ein Cross-Scripting zwischen einer SWF-Datei und mit ActionScript 3.0 geschriebenen SWF-Dateien aus anderen Domänen standardmäßig untersagt. Der Zugriff kann jedoch durch einen Aufruf der Methode `Security.allowDomain()` in der geladenen SWF-Datei ermöglicht werden. Weitere Informationen finden Sie unter „Cross-Scripting“ auf Seite 1138.

Standardmäßig gelten die folgenden allgemeinen Sicherheitsrichtlinien:

- Ressourcen in der gleichen Sicherheits-Sandbox können immer aufeinander zugreifen.
- Ausführbarer Code in Dateien in einer Remote-Sandbox kann nie auf lokale Dateien und Daten zugreifen.

In den Flash Player- und AIR-Laufzeitumgebungen gelten die folgenden Domänen als individuelle Domänen, für die jeweils einzelne Sicherheits-Sandboxen eingerichtet werden:

- `http://example.com`
- `http://www.example.com`

- `http://store.example.com`
- `https://www.example.com`
- `http://192.0.34.166`

Auch wenn eine benannte Domäne, wie http://example.com, einer bestimmten IP-Adresse zugeordnet werden kann (z. B. http://192.0.34.166), richten die Laufzeitumgebungen jeweils separate Sicherheits-Sandboxen ein.

Es gibt zwei allgemeine Methoden, über die der Entwickler einer SWF-Datei Zugriff auf Datenelemente gewähren kann, die sich in anderen Sandboxen als die SWF-Datei befinden:

- Die Methode `Security.allowDomain()` (siehe „Kontrolloptionen für Autoren (Entwickler)“ auf Seite 1129)
- Die URL-Richtliniendatei (siehe „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125)

In den Sicherheitsmodellen der Flash Player- und AIR-Laufzeitumgebungen wird zwischen dem Laden von Inhalten und dem Extrahieren oder Abrufen von Daten unterschieden. Als *Inhalt* werden Medien bezeichnet, darunter visuelle Medien, die die Laufzeitumgebungen anzeigen können, sowie Audio, Video oder SWF- oder HTML-Dateien mit angezeigten Medien. *Daten* sind Elemente, auf die definitionsgemäß nur über Code zugegriffen werden kann. Inhalte und Daten werden auf unterschiedliche Weise geladen.

- Laden von Inhalt: Sie können Inhalt mit Klassen wie Loader, Sound und NetStream laden, über MXML-Tags bei Verwendung von Flex oder über HTML-Tags in einer AIR-Anwendung.
- Extrahieren von Daten: Sie können Daten aus geladenen Medien mithilfe von Bitmap-Objekten, der Methoden `BitmapData.draw()` und `BitmapData.drawWithQuality()`, der Eigenschaft `Sound.id3` oder der Methode `SoundMixer.computeSpectrum()` extrahieren. Die `drawWithQuality`-Methode ist in Flash Player 11.3 und höher sowie AIR 3.3 und höher verfügbar.
- Zugreifen auf Daten: Sie können direkt auf Daten zugreifen, indem Sie diese mittels Klassen, wie URLStream, URLLoader, FileReference, Socket und XMLSocket, aus einer externen Datei (z. B. einer XML-Datei) laden. AIR bietet zusätzliche Klassen zum Laden von Daten, wie beispielsweise FileStream und XMLHttpRequest.

Das Flash Player-Sicherheitsmodell definiert unterschiedliche Richtlinien für das Laden von Inhalten und den Zugriff auf Daten. Im Allgemeinen gelten weniger Einschränkungen für das Laden von Inhalten als für den Zugriff auf Daten.

Normalerweise können Inhalte (SWF-Dateien, Bitmaps, MP3-Dateien und Videos) von allen Speicherorten geladen werden. Befindet sich der Inhalt jedoch in einer anderen Domäne als der ladende Code oder Inhalt, wird er in einer separaten Sicherheits-Sandbox partitioniert.

Für das Laden von Inhalten gelten nur wenige Einschränkungen:

- Standardmäßig werden lokale SWF-Dateien (Dateien, die nicht aus dem Netzwerk, sondern z. B. von der Festplatte des Benutzers geladen werden) in der „local-with-filesystem“-Sandbox (local-with-filesystem) klassifiziert. Dateien in dieser Sandbox können keine Inhalte aus dem Netzwerk laden. Weitere Informationen finden Sie unter „Lokale Sandboxen“ auf Seite 1117.
- Real-Time Messaging Protocol-Server (RTMP) können den Zugriff auf Inhalte einschränken. Weitere Informationen finden Sie unter „Über RTMP-Server bereitgestellte Inhalte“ auf Seite 1138.

Handelt es sich bei dem geladenen Medium um eine Grafik, Audio oder Video, kann eine SWF-Datei, die sich nicht in der gleichen Sicherheits-Sandbox befindet, nur auf die Daten (z. B. Pixel- und Sounddaten) zugreifen, wenn die Domäne dieser SWF-Datei in der Ursprungsdomäne der Medien in eine URL-Richtliniendatei aufgenommen wurde. Weitere Informationen finden Sie unter „Zugriff auf geladene Medien als Daten“ auf Seite 1142.

Weitere Formen von geladenen Daten sind z. B. Text- oder XML-Dateien, die mit einem URLLoader-Objekt geladen werden. Auch in diesem Fall muss der Zugriff auf Daten in einer anderen Sicherheits-Sandbox mittels einer URL-Richtliniendatei in der Ursprungsdomäne genehmigt werden. Weitere Informationen finden Sie unter „Verwenden von URLLoader und URLStream“ auf Seite 1145.

Hinweis: *Richtliniendateien sind grundsätzlich nicht erforderlich, damit Code, der in der AIR-Anwendungs-Sandbox ausgeführt wird, Remote-Inhalt oder -Daten laden kann.*

Sicherheits-Sandboxen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Clientcomputer können individuelle Dateien mit Code, Inhalt und Daten aus verschiedenen Quellen abrufen, wie beispielsweise externe Websites, lokale Dateisysteme oder installierte AIR-Anwendungen. Die Flash Player- und AIR-Laufzeitumgebungen weisen Codedateien und andere Ressourcen, wie gemeinsam genutzte Objekte, Bitmaps, Sounds, Videos und Datendateien, den Sicherheits-Sandboxen individuell zu, und zwar auf Grundlage ihres Ursprungs beim Laden. In den folgenden Abschnitten werden die von den Laufzeitumgebungen durchgesetzten Regeln beschrieben, die bestimmen, auf welche Elemente Code oder Inhalt, der in einer bestimmten Sandbox ausgeführt wird, zugreifen kann.

Weitere Informationen zur Flash Player-Sicherheit finden Sie im Abschnitt „Sicherheit“ des Flash Player Developer Center unter www.adobe.com/go/devnet_security_de.

Remote-Sandboxen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Flash Player- und AIR-Laufzeitumgebungen klassifizieren Elemente (einschließlich SWF-Dateien) aus dem Internet in separate Sandboxes, die der jeweiligen Ursprungsdomäne entsprechen. Beispielsweise werden Elemente, die aus *example.com* geladen werden, in einer anderen Sicherheits-Sandbox platziert als Elemente, die aus *foo.org* geladen werden. Standardmäßig ist diesen Dateien der Zugriff auf andere Ressourcen auf ihrem eigenen Server gestattet. Remote-SWF-Dateien kann über explizite Website- und Autorberechtigungen, z. B. URL-Richtliniendateien und die Methode `Security.allowDomain()`, der Zugriff auf weitere Daten aus anderen Domänen gestattet werden. Weitere Informationen finden Sie unter „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125 und „Kontrolloptionen für Autoren (Entwickler)“ auf Seite 1129.

Remote-SWF-Dateien können keine lokalen Daten oder Ressourcen laden.

Weitere Informationen zur Flash Player-Sicherheit finden Sie im Abschnitt „Sicherheit“ des Flash Player Developer Center unter www.adobe.com/go/devnet_security_de.

Lokale Sandboxen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Als *lokale Datei* werden alle Dateien bezeichnet, die über das Protokoll `file:` oder einen UNC-Pfad (Universal Naming Convention) referenziert werden. Lokale SWF-Dateien befinden sich in einer der vier folgenden lokalen Sandboxen:

- Lokale Sandbox mit Dateisystemzugriff (local-with-filesystem) – Aus Sicherheitsgründen platzieren die Flash Player- und AIR-Laufzeitumgebungen alle lokalen Dateien standardmäßig in der Sandbox „local-with-file-system“. Ausführbarer Code in dieser Sandbox kann lokale Dateien lesen (z. B. mit der URLLoader-Klasse). Die Kommunikation mit dem Netzwerk ist jedoch nicht möglich. Der Benutzer kann dann sicher sein, dass die lokalen Daten nicht in das Netzwerk geraten oder auf andere Weise unbefugt freigegeben werden.

- Lokale Sandbox mit Netzwerkzugriff (local-with-networking) – Beim Kompilieren einer SWF-Datei können Sie angeben, ob diese beim Ausführen als lokale Datei über Netzwerkzugriff verfügt (siehe „Einstellen des Sandbox-Typs von lokalen SWF-Dateien“ auf Seite 1120). Diese Dateien werden in der „local-with-networking“-Sandbox platziert. Wenn SWF-Dateien der „local-with-networking“-Sandbox zugewiesen sind, ist kein lokaler Dateizugriff möglich. Dafür können die SWF-Dateien auf Daten aus dem Netzwerk zugreifen. SWF-Dateien in der „local-with-networking“-Sandbox sind jedoch nur dann berechtigt, aus dem Netzwerk abgeleitete Daten zu lesen, wenn durch eine URL-Richtliniendatei oder den Aufruf der Methode `Security.allowDomain()` Zugriffsrechte für diesen Vorgang erteilt wurden. Für diese Zugriffsrechte muss eine URL-Richtliniendatei den Zugriff auf *alle* Domänen mittels`<allow-access-from domain="*"/>` oder `Security.allowDomain("*")` gewähren. Weitere Informationen finden Sie unter „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125 und „Kontrolloptionen für Autoren (Entwickler)“ auf Seite 1129.
- Lokale vertrauenswürdige Sandbox (local-trusted) – Lokale SWF-Dateien, die als vertrauenswürdig eingestuft sind (durch den Benutzer oder Installationsprogramme), werden in der „local-trusted“-Sandbox abgelegt. Systemadministratoren und Benutzer können eine lokale SWF-Datei auch basierend auf Sicherheitsüberlegungen in eine „local-trusted“-Sandbox verschieben oder daraus entfernen (siehe „Kontrolloptionen für Administratoren“ auf Seite 1122 und „Kontrolloptionen für Benutzer“ auf Seite 1124). SWF-Dateien, die der „local-trusted“-Sandbox zugewiesen sind, können mit allen anderen SWF-Dateien interagieren und von allen Speicherorten Daten laden (remote oder lokal).
- Die Sandbox der AIR-Anwendung – Diese Sandbox enthält Inhalte, die zusammen mit der ausgeführten AIR-Anwendung installiert wurden. Standardmäßig kann Code, der in der Sandbox der AIR-Anwendung ausgeführt wird, auf Code aus einer beliebigen Domäne verweisen (Cross-Scripting). Dateien, die nicht in der Sandbox der AIR-Anwendung enthalten sind, können jedoch nicht per Cross-Scripting auf Code in der Anwendungs-Sandbox verweisen. Standardmäßig können Code und Inhalte in der Sandbox der AIR-Anwendung Inhalte und Daten aus einer beliebigen Domäne laden.

Der Datenaustausch zwischen den Sandboxen „local-with-networking“ und „local-with-filesystem“ sowie zwischen der Sandbox „local-with-filesystem“ und Remote-Sandboxen ist streng verboten. Eine Genehmigung für einen solchen Datenaustausch kann weder von einer in Flash Player ausgeführten Anwendung, noch durch einen Benutzer oder Administrator erteilt werden.

Cross-Scripting zwischen lokalen HTML-Dateien und lokalen SWF-Dateien, z. B. mit der ExternalInterface-Klasse, ist nur dann möglich, wenn sich sowohl die HTML- als auch die SWF-Datei in der „local-trusted“-Sandbox befinden. Der Grund hierfür ist, dass sich die lokalen Sicherheitsmodelle für Browser vom lokalen Sicherheitsmodell von Flash Player unterscheiden.

SWF-Dateien in der „local-with-networking“-Sandbox können keine SWF-Dateien in der „local-with-filesystem“-Sandbox laden. SWF-Dateien in der „local-with-networking“-Sandbox können keine SWF-Dateien in der „local-with-filesystem“-Sandbox laden.

Die AIR-Anwendungs-Sandbox

Adobe AIR 1.0 und höher

Die Adobe AIR-Laufzeitumgebung erweitert das Sicherheits-Sandbox-Modell von Flash Player um eine zusätzliche Sandbox, die sogenannte *Anwendungs*-Sandbox. Dateien, die als Teil einer AIR-Anwendung installiert wurden, werden in die Anwendungs-Sandbox geladen. Für alle anderen von der Anwendung geladenen Dateien gelten Sicherheitseinschränkungen gemäß den Richtlinien des regulären Flash Player-Sicherheitsmodells.

Wenn eine Anwendung installiert wird, werden alle Dateien, die im AIR-Paket enthalten sind, in einem Anwendungsverzeichnis auf dem Computer des Benutzers installiert. Entwickler können im Code über das `app:/`-URL-Schema (siehe „URI-Schemas“ auf Seite 867) auf dieses Verzeichnis verweisen. Alle Dateien in der Anwendungsverzeichnisstruktur werden der Anwendungs-Sandbox zugewiesen, wenn die Anwendung ausgeführt wird. Inhalt in der Anwendungs-Sandbox ist mit allen Berechtigungen versehen, die für eine AIR-Anwendung verfügbar sind, darunter auch für die Interaktion mit dem lokalen Dateisystem.

Viele AIR-Anwendungen verwenden nur diese lokal installierten Dateien, um die Anwendung auszuführen. AIR-Anwendungen sind jedoch nicht auf die Dateien im Anwendungsverzeichnis beschränkt; sie können jeden Dateityp von jeder beliebigen Quelle laden. Dazu gehören lokale Dateien auf dem Computer des Benutzers sowie Dateien aus verfügbaren externen Quellen, zum Beispiel von einem lokalen Netzwerk oder aus dem Internet. Der Dateityp hat keinen Einfluss auf Sicherheitsbeschränkungen; geladene HTML-Dateien haben dieselben Sicherheitsberechtigungen wie geladene SWF-Dateien aus derselben Quelle.

Inhalt in der Anwendungs-Sandbox hat Zugriff auf AIR-APIs, auf die Inhalt in anderen Sandboxen nicht zugreifen kann. Zum Beispiel ist die `air.NativeApplication.nativeApplication.applicationDescriptor`-Eigenschaft, die den Inhalt der Anwendungsdeskriptordatei für eine Anwendung zurückgibt, auf den Inhalt in der Sicherheits-Sandbox der Anwendung beschränkt. Ein weiteres Beispiel einer beschränkten API ist die FileStream-Klasse, die Methoden für das Lesen und Schreiben im lokalen Dateisystem enthält.

ActionScript-APIs, die nur für Inhalt in der Sicherheits-Sandbox der Anwendung zur Verfügung stehen, sind im *ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform* mit dem AIR-Logo gekennzeichnet. Wenn Sie diese APIs in anderen Sandboxen verwenden, gibt die Laufzeitumgebung eine SecurityError-Ausnahme aus.

Bei HTML-Inhalt (in einem HTMLLoader-Objekt) sind alle AIR-JavaScript-APIs (diejenigen, die über die `window.runtime`-Eigenschaft verfügbar sind, oder über das `air`-Objekt, wenn die Datei „AIRAliases.js“ verwendet wird) für Inhalt in der Anwendungs-Sandbox verfügbar. HTML-Inhalt in anderen Sandboxen hat keinen Zugriff auf die `window.runtime`-Eigenschaft, deshalb kann dieser Inhalt nicht auf die AIR- oder Flash Player-APIs zugreifen.

Für Inhalt, der in der AIR-Anwendungs-Sandbox ausgeführt wird, gelten die folgenden zusätzlichen Einschränkungen:

- Für HTML-Inhalt in der Anwendungs-Sandbox gibt es Beschränkungen bezüglich der Verwendung von APIs, die Strings dynamisch in ausführbaren Code umwandeln, nachdem der Code geladen wurde. Auf diese Weise wird verhindert, dass die Anwendung ungewollt Code aus anderen Quellen als der Anwendung (zum Beispiel von potenziell unsicheren Netzwerkdomänen) einfügt und ausführt. Ein Beispiel ist die Verwendung der `eval()`-Funktion. Ausführliche Informationen finden Sie unter „Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen“ auf Seite 1161.
- Um mögliche Phishing-Angriffe zu verhindern, werden `img`-Tags in HTML-Inhalten in ActionScript-TextField-Objekten in SWF-Inhalten in der Anwendungs-Sandbox ignoriert.
- Inhalt in der Anwendungs-Sandbox kann das `asfunction`-Protokoll in HTML-Inhalt in ActionScript 2.0-Textfeldern nicht verwenden.
- SWF-Inhalt in der Anwendungs-Sandbox kann den domänenübergreifenden Cache nicht nutzen. Diese Funktion wurden mit Flash Player 9 Update 3 eingeführt. Damit kann Flash Player Adobe-Plattformkomponenteninhalt zwischenspeichern und bei Bedarf in geladenen SWF-Inhalten wiederverwenden (sodass der Inhalt nicht mehrmals neu geladen werden muss).

Einschränkungen für JavaScript in AIR

Adobe AIR 1.0 und höher

Anders als Inhalt in der Anwendungs-Sandbox *kann* JavaScript-Inhalt in einer anwendungsfremden Sandbox jederzeit die `eval()`-Funktion aufrufen, um dynamisch generierten Code auszuführen. Es gelten jedoch Einschränkungen für JavaScript, das in einer anwendungsfremden Sicherheits-Sandbox innerhalb von AIR ausgeführt wird. Dazu gehört Folgendes:

- JavaScript-Code in einer anwendungsfremden Sandbox hat keinen Zugriff auf das `window.runtime`-Objekt und der Code kann keine AIR-APIs ausführen.
- Standardmäßig kann Inhalt in einer anwendungsfremden Sandbox keine XMLHttpRequest-Aufrufe verwenden, um Daten aus anderen Domänen als der Domäne, die die Anforderung aufruft, zu laden. Anwendungscode kann anwendungsfremden Inhalten jedoch die Berechtigung dazu gewähren, indem ein `allowCrossdomainXHR`-Attribut im Frame oder Inlineframe mit dem Inhalt gesetzt wird. Weitere Informationen finden Sie unter „Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen“ auf Seite 1161.
- Es gibt Beschränkungen für das Aufrufen der JavaScript-Methode `window.open()`. Weitere Informationen finden Sie unter „Beschränkungen beim Aufrufen der JavaScript-window.open()-Methode“ auf Seite 1164.
- HTML-Inhalte in Remote-Sandboxen (Netzwerk-Sandboxen) können nur CSS-, `frame`-, `iframe`- und `img`-Inhalt aus Remotedomänen (von Netzwerk-URLs) laden.
- HTML-Inhalt in „local-with-filesystem“-, „local-with-networking“- oder „local-trusted“-Sandboxen kann nur CSS-, `frame`-, `iframe`- und `img`-Inhalte aus lokalen Sandboxen (nicht aus der Anwendung oder aus Netzwerk-URLs) laden.

Ausführliche Informationen finden Sie unter „Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen“ auf Seite 1161.

Einstellen des Sandbox-Typs von lokalen SWF-Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Ein Endbenutzer oder der Administrator eines Computers kann festlegen, dass eine lokale SWF-Datei als vertrauenswürdig eingestuft wird und ihr somit das Laden von Daten aus allen Domänen (lokal und remote) gestatten. Dies wird in den Verzeichnissen „Global Flash Player Trust“ und „User Flash Player Trust“ festgelegt. Weitere Informationen finden Sie unter „Kontrolloptionen für Administratoren“ auf Seite 1122 und „Kontrolloptionen für Benutzer“ auf Seite 1124.

Weitere Informationen zu lokalen Sandboxen finden Sie unter „Lokale Sandboxen“ auf Seite 1117.

Adobe Flash Professional

Sie können eine SWF-Datei so konfigurieren, dass sie entweder der „local-with-filesystem“-Sandbox oder der „local-with-networking“-Sandbox zugewiesen wird, indem Sie die Veröffentlichungseinstellungen des Dokuments im Authoring-Tool festlegen.

Adobe Flex
Sie können eine SWF-Datei so konfigurieren, dass sie entweder der „local-with-filesystem"-Sandbox oder der „local-with-networking"-Sandbox zugewiesen wird, indem Sie das `use-network`-Flag im Adobe Flex-Compiler festlegen. Weitere Informationen finden Sie unter „About the application compiler options" im Handbuch *Building and Deploying Adobe Flex 3 Applications*.

Security.sandboxType-Eigenschaft

Flash Player 9 und höher, Adobe AIR 1.0 und höher

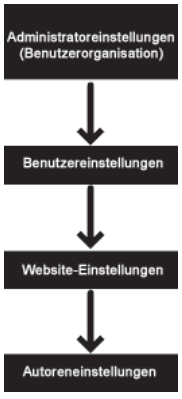
Ein Autor einer SWF-Datei kann mit der schreibgeschützten statischen `Security.sandboxType`-Eigenschaft den Sandbox-Typ ermitteln, dem die Flash Player- oder AIR-Laufzeitumgebung die SWF-Datei zugewiesen hat. Die Security-Klasse enthält Konstanten, die mögliche Werte der Eigenschaft `Security.sandboxType` darstellen:

- `Security.REMOTE` – Die SWF-Datei stammt von einer Internet-URL und kann gemäß den domänenbasierten Sandbox-Regeln verwendet werden.
- `Security.LOCAL_WITH_FILE` – Die SWF-Datei ist eine lokale Datei, die der Benutzer nicht als vertrauenswürdig eingestuft hat und die nicht für die Verwendung im Netzwerk veröffentlicht wurde. Die SWF-Datei kann lokale Datenquellen lesen, aber keine Verbindung mit dem Internet herstellen.
- `Security.LOCAL_WITH_NETWORK` – Die SWF-Datei ist eine lokale Datei, die der Benutzer als nicht vertrauenswürdig eingestuft hat und die für die Verwendung im Netzwerk veröffentlicht wurde. Die SWF-Datei kann eine Verbindung mit dem Internet herstellen, jedoch keine lokalen Datenquellen lesen.
- `Security.LOCAL_TRUSTED` – Die SWF-Datei ist eine lokale Datei, die der Benutzer über den Einstellungsmanager oder eine Flash Player Trust-Konfigurationsdatei als vertrauenswürdig eingestuft hat. Die SWF-Datei kann lokale Datenquellen lesen und eine Verbindung mit dem Internet herstellen.
- `Security.APPLICATION` – Die SWF-Datei wird in einer AIR-Anwendung ausgeführt und wurde mit dem Paket (der AIR-Datei) für diese Anwendung installiert. Standardmäßig können Dateien in der Sandbox der AIR-Anwendung auf Dateien aus einer beliebigen Domäne verweisen (Cross-Scripting). Dateien, die nicht in der Sandbox der AIR-Anwendung enthalten sind, können jedoch nicht per Cross-Scripting auf die AIR-Datei verweisen. Standardmäßig können Dateien in der Sandbox der AIR-Anwendung Inhalte und Daten aus einer beliebigen Domäne laden.

Steuerung der Berechtigungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Sicherheitsmodell von Flash Player zur Clientlaufzeit wurde um Ressourcen konzipiert, bei denen es sich um Objekte handelt, wie SWF-Dateien, lokale Daten und Internet-URLs. *Stakeholders* (Interessengruppen) sind die Eigentümer oder die Benutzer dieser Ressourcen. Stakeholders haben die Kontrolle (Sicherheitseinstellungen) über ihre eigenen Ressourcen und jeder Ressource sind vier Stakeholders zugeordnet. Flash Player erzwingt eine strikte Autoritätshierarchie für diese Kontrollen. Dies wird in der folgenden Abbildung verdeutlicht:



Hierarchie der Sicherheitskontrollen

Dies bedeutet beispielsweise, dass die Einschränkung des Zugriffs auf eine Ressource durch einen Administrator von keinem anderen Stakeholder außer Kraft gesetzt werden kann.

Bei AIR-Anwendungen gelten diese Berechtigungssteuerungen nur für Inhalt, der außerhalb der AIR-Anwendungs-Sandbox ausgeführt wird.

Kontrolloptionen für Administratoren

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Der Administrator eines Computers (ein Benutzer, der sich mit Administratorrechten angemeldet hat) kann Flash Player-Sicherheitseinstellungen anwenden, die sich auf alle Benutzer des Computers auswirken. In einer privaten Umgebung, beispielsweise bei einem Heimcomputer, gibt es in der Regel nur einen Benutzer, der somit auch administrativen Zugriff hat. Auch in einer Unternehmensumgebung können bestimmte Benutzer Administratorrechte auf einem Computer haben.

Es gibt zwei Arten von Kontrollmöglichkeiten für Administratoren:

- Die Datei „mms.cfg“
- Das Global Flash Player Trust-Verzeichnis

Die Datei „mms.cfg“

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Datei „mms.cfg“ ist eine Textdatei, in der Administratoren den Zugriff auf verschiedene Funktionen erteilen oder einschränken können. Beim Starten liest Flash Player die Sicherheitseinstellungen in dieser Datei und verwendet sie zum Einschränken von Funktionsmerkmalen. Die Datei „mms.cfg“ umfasst Einstellungen, mit denen Administratoren Dinge wie Sicherheitseinstellungen, die lokale Dateisicherheit, Socketverbindungen usw. verwalten können.

Eine SWF-Datei kann auf einige Informationen zu den Funktionen zugreifen, die durch Aufrufen der Eigenschaften `Capabilities.avHardwareDisable` und `Capabilities.localFileReadDisable` deaktiviert wurden. Jedoch können die meisten Einstellungen in der Datei „mms.cfg“ nicht mit ActionScript abgefragt werden.

Um von der Anwendung unabhängige Richtlinien für Sicherheit und Privatsphäre eines Computers durchzusetzen, darf die Datei „mms.cfg“ nur von Systemadministratoren geändert werden. Die Datei „mms.cfg“ wird nicht von den Installationsprogrammen von Anwendungen verwendet. Obwohl ein Installationsprogramm, das mit den Administratorrechten ausgeführt wird, den Inhalt der Datei „mms.cfg“ modifizieren könnte, betrachtet Adobe eine solche Nutzung als Vertrauensbruch gegenüber dem Benutzer und rät den Entwicklern von Installationsprogrammen, die Datei „mms.cfg“ niemals zu modifizieren.

Die Datei „mms.cfg“ wird unter folgendem Pfad gespeichert:

- Windows: *system*\Macromed\Flash\mms.cfg

  (beispielsweise C:\WINDOWS\system32\Macromed\Flash\mms.cfg)

- Mac: *app support*/Macromedia/mms.cfg

  (beispielsweise /Library/Application Support/Macromedia/mms.cfg)

Weitere Informationen zu der Datei „mms.cfg“ finden Sie im Administratorhandbuch zu Flash Player unter www.adobe.com/go/flash_player_admin_de.

Das Global Flash Player Trust-Verzeichnis

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Benutzer mit Administratorrechten und Installationsprogramme können bestimmte lokale SWF-Dateien als vertrauenswürdig für alle Benutzer einstufen. Diese SWF-Dateien werden der lokalen vertrauenswürdigen Sandbox (local-trusted) zugeordnet. Sie können mit allen anderen SWF-Dateien interagieren und von allen Speicherorten (remote oder lokal) Daten laden. Dateien werden im Global Flash Player Trust-Verzeichnis unter folgendem Pfad als vertrauenswürdig eingestuft:

- Windows: *system*\Macromed\Flash\FlashPlayerTrust

  (beispielsweise C:\WINDOWS\system32\Macromed\Flash\FlashPlayerTrust)

- Mac: *app support*/Macromedia/FlashPlayerTrust

  (beispielsweise /Library/Application Support/Macromedia/FlashPlayerTrust)

Das Flash Player Trust-Verzeichnis kann eine beliebige Anzahl von Textdateien enthalten, die jeweils vertrauenswürdige Pfade (ein Pfad pro Zeile) enthalten. Jeder Pfad kann eine einzelne SWF-Datei, eine HTML-Datei oder ein Verzeichnis sein. Kommentarzeilen beginnen mit dem #-Symbol. Eine Flash Player Trust-Konfigurationendatei mit dem folgenden Text stuft beispielsweise alle Dateien im angegebenen Verzeichnis und allen Unterverzeichnissen als vertrauenswürdig ein:

```
# Trust files in the following directories:
C:\Documents and Settings\All Users\Documents\SampleApp
```

Die in einer Trust-Konfigurationsdatei aufgeführten Pfade müssen immer lokale Pfade oder SMB-Netzwerkpfade sein. Ein HTTP-Pfad in einer Trust-Konfigurationsdatei wird ignoriert, denn es können nur lokale Dateien als vertrauenswürdig eingestuft werden.

Um Konflikte zu vermeiden, benennen Sie jede Trust-Konfigurationsdatei entsprechend der installierten Anwendung und verwenden Sie die Dateinamenserweiterung „.cfg“.

Als Entwickler, der eine lokal ausgeführte SWF-Datei über ein Installationsprogramm verteilt, können Sie mithilfe des Installationsprogramms eine Konfigurationsdatei in das Global Flash Player Trust-Verzeichnis einfügen und somit der von Ihnen verteilten Datei alle notwendigen Rechte erteilen. Dazu muss das Installationsprogramm von einem Benutzer mit Administratorrechten ausgeführt werden. Im Gegensatz zur Datei „mms.cfg“ darf das Global Flash Player Trust-Verzeichnis verwendet werden, um Installationsprogrammen vertrauenswürdige Zugriffsrechte zu erteilen. Sowohl Benutzer mit Administratorrechten als auch Installationsprogramme können mit dem Global Flash Player Trust-Verzeichnis vertrauenswürdige lokale Anwendungen zuweisen.

Es gibt auch Flash Player Trust-Verzeichnisse für einzelne Benutzer (siehe „Kontrolloptionen für Benutzer“ auf Seite 1124).

Kontrolloptionen für Benutzer

Flash Player 9 und höher

Flash Player bietet drei Mechanismen zum festlegen von Berechtigungen für verschiedene Benutzerebenen: die Einstellungs-UI, den Einstellungsmanager und das User Flash Player Trust-Verzeichnis.

Einstellungs-UI und Einstellungsmanager

Flash Player 9 und höher

Die Einstellungs-UI ist ein schneller, interaktiver Mechanismus zum Konfigurieren der Einstellungen für eine bestimmte Domäne. Der Einstellungsmanager weist eine wesentlich detailliertere Schnittstelle auf und bietet die Möglichkeit, globale Änderungen vorzunehmen, die sich auf die Zugriffsrechte vieler oder aller Domänen auswirken. Wenn eine SWF-Datei eine neue Berechtigung anfordert, die Laufzeitentscheidungen hinsichtlich der Sicherheit oder Privatsphäre erfordert, werden Dialogfelder angezeigt, in denen Benutzer bestimmte Flash Player-Einstellungen anpassen können.

Der Einstellungsmanager und die Einstellungs-UI bieten sicherheitsbezogene Optionen wie Kamera- und Mikrofoneinstellungen, gemeinsame Objektspeichereinstellungen, Einstellungen für ältere Inhalte usw. Weder der Einstellungsmanager noch die Einstellungs-UI stehen für AIR-Anwendungen zur Verfügung.

Hinweis: *Einstellungen, die Sie in der Datei „mms.cfg“ vornehmen (siehe „Kontrolloptionen für Administratoren“ auf Seite 1122), werden im Einstellungsmanager nicht widergespiegelt.*

Weitere Informationen zum Einstellungsmanager finden Sie unter www.adobe.com/go/settingsmanager_de.

User Flash Player Trust-Verzeichnis

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Benutzer und Installationsprogramme können bestimmte lokale SWF-Dateien als vertrauenswürdig einstufen. Diese SWF-Dateien werden der lokalen vertrauenswürdigen Sandbox (local-trusted) zugeordnet. Sie können mit allen anderen SWF-Dateien interagieren und von allen Speicherorten (remote oder lokal) Daten laden. Ein Benutzer stuft eine Datei im User Player Trust-Verzeichnis als vertrauenswürdig ein. Dies ist das gleiche Verzeichnis wie der Speicherbereich für gemeinsam verwendete Flash-Objekte an den folgenden Speicherorten (die Speicherorte gelten für den aktuellen Benutzer):

- Windows: app data\Macromedia\Flash Player\#Security\FlashPlayerTrust

  (beispielsweise C:\Dokumente und Einstellungen\<Benutzername>\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust unter Windows XP oder C:\Users\<Benutzername>\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust unter Windows Vista)

  In Windows ist der Ordner „Application Data“ standardmäßig ausgeblendet. Klicken Sie zum Anzeigen ausgeblendeter Ordner und Dateien auf „Arbeitsplatz“, um den Windows-Explorer zu öffnen. Wählen Sie dann „Extras“ > „Ordneroptionen“ und anschließend die Registerkarte „Ansicht“ aus. Aktivieren Sie auf der Registerkarte „Ansicht“ das Kontrollkästchen „Alle Dateien und Ordner anzeigen“.

- Mac: app data/Macromedia/Flash Player/#Security/FlashPlayerTrust

  (beispielsweise /Users/<Benutzername>/Library/Preferences/Macromedia/Flash Player/#Security/FlashPlayerTrust)

  Diese Einstellungen wirken sich nur auf den aktuellen Benutzer aus, nicht auf andere Benutzer, die sich am Computer anmelden. Installiert ein Benutzer ohne Administratorrechte eine Anwendung in seinem eigenen Bereich des Systems, lässt das User Flash Player Trust-Verzeichnis das Installationsprogramm die Anwendung für diesen Benutzer als vertrauenswürdig registrieren.

  Als Entwickler, der eine lokal ausgeführte SWF-Datei über ein Installationsprogramm verteilt, können Sie über das Installationsprogramm eine Konfigurationsdatei in das User Flash Player Trust-Verzeichnis einfügen und somit der von Ihnen verteilten Datei alle notwendigen Rechte erteilen. Auch in dieser Situation wird das User Flash Player Trust-Verzeichnis als eine Kontrolloption für den Benutzer gewertet, da es von einem Benutzerereignis (Installation) initialisiert wird.

  Weiterhin gibt es ein Global Flash Player Trust-Verzeichnis, das von Benutzern mit Administratorrechten oder von Installationsprogrammen verwendet wird, um eine Anwendung für alle Benutzer eines Computers zu registrieren (siehe „Kontrolloptionen für Administratoren“ auf Seite 1122).

Kontrolloptionen für Websites (Richtliniendateien)

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um Daten auf einem Webserver auch SWF-Dateien in anderen Domänen zur Verfügung zu stellen, können Sie eine Richtliniendatei auf dem Server erstellen. Eine *Richtliniendatei* ist eine XML-Datei, die in ein bestimmtes Verzeichnis auf dem Server eingefügt wird.

Richtliniendateien steuern den Zugriff auf verschiedene Datenelemente, einschließlich der Folgenden:

- Daten in Bitmaps, Sounds und Videos
- Laden von XML- und Textdateien

- Importieren von SWF-Dateien aus anderen Sicherheitsdomänen in die Sicherheitsdomäne der ladenden SWF-Datei
- Zugriff auf Socket- und XML-Socketverbindungen

ActionScript-Objekte instanziieren zwei Arten von Serververbindungen: dokumentbasierte Serververbindungen und Socketverbindungen. ActionScript-Objekte wie Loader, Sound, URLLoader und URLStream instanziieren dokumentbasierte Serververbindungen. Diese Objekte laden eine Datei von einer URL. ActionScript-Socket- und XMLSocket-Objekte stellen Socketverbindungen her, die mit Streaming-Daten anstelle von geladenen Dokumenten arbeiten.

Da Flash Player zwei Arten von Serververbindungen unterstützt, gibt es auch zwei Arten von Richtliniendateien: URL-Richtliniendateien und Socket-Richtliniendateien.

- Dokumentbasierte Verbindungen erfordern *URL-Richtliniendateien*. In diesen Dateien wird angegeben, dass die Daten und Dokumente auf dem Server für SWF-Dateien aus bestimmten oder allen Domänen verfügbar sind.
- Socketverbindungen erfordern *Socket-Richtliniendateien*, die mithilfe der Socket- und XMLSocket-Klassen einen direkten Netzwerkbetrieb auf der unteren TCP-Socketebene ermöglichen.

Flash Player setzt voraus, dass die Richtliniendateien mit dem gleichen Protokoll übertragen werden, das von der gewünschten Verbindung verwendet wird. Wenn Sie eine Richtliniendatei auf Ihrem HTTP-Server platzieren, können SWF-Dateien aus anderen Domänen Daten von diesem Server als HTTP-Server laden. Wenn Sie jedoch keine Socket-Richtliniendatei auf dem gleichen Server bereitstellen, verbieten Sie SWF-Dateien aus anderen Domänen, eine Verbindung auf Socketebene mit dem Server herzustellen. Anders ausgedrückt, das Verfahren zum Abrufen einer Richtliniendatei muss dem Verbindungsverfahren entsprechen.

In diesem Abschnitt werden Verwendung und Syntax der Richtliniendatei in Bezug auf SWF-Dateien, die für Flash Player 10 veröffentlicht wurden, kurz erläutert. (Die Implementierung von Richtliniendateien in früheren Flash Player-Versionen weicht geringfügig hiervon ab, da in den neueren Versionen die Flash Player-Sicherheit erhöht wurde.) Weitere Informationen zu Richtliniendateien finden Sie im Abschnitt „Richtliniendateiänderungen in Flash Player 9“ des Flash Player Developer Center unter www.adobe.com/go/devnet_security_de.

In der AIR-Anwendungs-Sandbox ausgeführter Code benötigt keine Richtliniendatei für den Zugriff auf Daten von URLs oder Sockets. Für Code, der in der AIR-Anwendung in einer anwendungsfremden Sandbox ausgeführt wird, ist jedoch eine Richtliniendatei erforderlich.

Master-Richtliniendateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player (und AIR-Inhalte, die sich nicht in der AIR-Anwendungs-Sandbox befinden) suchen zuerst im Stammverzeichnis des Servers nach einer URL-Richtliniendatei mit dem Namen `crossdomain.xml` und dann auf Port 843 nach einer Socket-Richtliniendatei. Eine Datei in einem dieser beiden Speicherorte wird als *Master-Richtliniendatei* bezeichnet. (Im Falle von Socketverbindungen sucht Flash Player zudem nach einer Socket-Richtliniendatei am gleichen Port wie die Hauptverbindung. Eine an diesem Port gefundene Richtliniendatei gilt jedoch nicht als Master-Richtliniendatei.)

Eine Master-Richtliniendatei kann nicht nur Zugriffsrechte festlegen, sondern auch eine *meta-policy*-Anweisung enthalten. Eine „meta-policy“-Anweisung gibt an, in welchen Speicherorten Richtliniendateien enthalten sein können. Die standardmäßige „meta-policy“-Anweisung für URL-Richtliniendateien ist „master-only“, d. h. auf dem Server ist ausschließlich die Richtliniendatei „/crossdomain.xml“ zulässig. Die standardmäßige „meta-policy“-Anweisung für Socket-Richtliniendateien ist „all“, d. h. jeder Socket auf dem Host kann als Socket-Richtliniendatei verwendet werden.

Hinweis: *In Flash Player 9 und früheren Versionen war die standardmäßige „meta-policy"-Anweisung für URL-Richtliniendateien „all", sodass alle Verzeichnisse eine Richtliniendatei enthalten konnten. Wenn Sie Anwendungen bereitgestellt haben, die Richtliniendateien aus anderen Speicherorten als der Standarddatei „/crossdomain.xml" laden, und diese Anwendungen nun in Flash Player 10 ausgeführt werden, müssen Sie (oder der Serveradministrator) die Master-Richtliniendatei so bearbeiten, dass zusätzliche Richtliniendateien zulässig sind. Weitere Informationen zur Angabe eines anderen „meta-policy"-Ausdrucks finden Sie im Abschnitt „Richtliniendateiänderungen in Flash Player 9" des Flash Player Developer Center unter www.adobe.com/go/devnet_security_de.*

Eine SWF-Datei kann durch Aufrufen der Methode `Security.loadPolicyFile()` auch auf einen anderen Richtliniendateinamen oder ein anderes Verzeichnis prüfen. Wenn die Master-Richtliniendatei jedoch nicht festlegt, dass das Zielverzeichnis Richtliniendateien bereitstellen kann, hat der Aufruf der Methode `loadPolicyFile()` keine Auswirkung, selbst wenn in diesem Verzeichnis eine Richtliniendatei vorhanden ist. Rufen Sie die Methode `loadPolicyFile()` auf, bevor Sie Netzwerkoperationen durchführen, die die Richtliniendatei voraussetzen. Flash Player fügt Netzwerkanforderungen automatisch zur Warteschlange hinter den entsprechenden Richtliniendateiversuchen hinzu. Es ist daher zulässig, die Methode `Security.loadPolicyFile()` unmittelbar vor dem Initiieren einer Netzwerkoperation aufzurufen.

Beim Überprüfen auf eine Master-Richtliniendatei wartet Flash Player drei Sekunden lang auf eine Antwort des Servers. Bleibt die Antwort aus, geht Flash Player davon aus, dass keine Master-Richtliniendatei vorhanden ist. Für Aufrufe der Methode `loadPolicyFile()` besteht hingegen kein standardmäßiges Zeitlimit. Flash Player geht davon aus, dass die aufgerufene Datei vorhanden ist, und wartet für einen beliebigen Zeitraum, um diese zu laden. Um sicherzustellen, dass die Master-Richtliniendatei geladen wird, sollten Sie diese daher mit der Methode `loadPolicyFile()` explizit aufrufen.

Der Name der Methode lautet zwar `Security.loadPolicyFile()`, die Richtliniendatei wird jedoch erst geladen, wenn ein Netzwerkaufruf, der eine Richtliniendatei voraussetzt, ausgegeben wird. Der Aufruf der Methode `loadPolicyFile()` informiert Flash Player lediglich, wo Richtliniendateien bei Bedarf gesucht werden sollen.

Sie erhalten keine Benachrichtigung, wenn eine Richtliniendateianforderungen initiiert oder abgeschlossen wurde. Hierzu besteht auch keine Notwendigkeit. Flash Player führt Richtlinienüberprüfungen asynchron durch und wartet mit der Initiierung von Verbindungen automatisch, bis die Richtliniendateiüberprüfungen erfolgreich waren.

Die Informationen in den folgenden Abschnitten beziehen sich ausschließlich auf URL-Richtliniendateien. Weitere Informationen zu Socket-Richtliniendateien finden Sie unter „Herstellen einer Verbindung mit Sockets" auf Seite 1145.

Umfang einer URL-Richtliniendatei

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine URL-Richtliniendatei gilt nur für das Verzeichnis, aus dem sie geladen wurde, sowie dessen Unterverzeichnisse. Eine Richtliniendatei im Stammverzeichnis gilt für den gesamten Server, während eine Richtliniendatei, die aus einem anderen Verzeichnis aufgerufen wurde, nur für dieses Verzeichnis und die zugehörigen Unterverzeichnisse gilt.

Eine Richtliniendatei wirkt sich nur auf den Zugriff auf den Server aus, auf dem sie gespeichert ist. Eine Richtliniendatei unter „https://www.adobe.com:8080/crossdomain.xml" gilt beispielsweise nur für Aufrufe zum Laden von Daten, die an www.adobe.com per HTTPS an Port 8080 gerichtet wurden.

Festlegen von Zugriffsrechten in einer URL-Richtliniendatei

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Eine Richtliniendatei enthält ein einziges `<cross-domain-policy>`-Tag, das wiederum keine oder mehrere `<allow-access-from>`-Tags enthält. Jedes `<allow-access-from>`-Tag enthält das Attribut `domain`, das entweder eine genaue IP-Adresse, eine genaue Domäne oder eine Platzhalterdomäne (beliebige Domäne) festlegt. Platzhalterdomänen können auf zwei Arten angegeben werden:

- Durch ein einzelnes Sternchen (*) für alle Domänen und IP-Adresse
- Durch ein Sternchen gefolgt von einem Suffix für die Domänen, die auf das angegebene Suffix enden

Suffixe müssen mit einem Punkt beginnen. Platzhalterdomänen mit Suffixen können jedoch für Domänen stehen, die nur das Suffix, nicht jedoch den Punkt enthalten. Die Domäne „xyz.com“ gilt beispielsweise Bestandteil von „*.xyz.com“. In IP-Domänenspezifikationen sind keine Platzhalter zugelassen.

Im folgenden Beispiel ist eine URL-Richtliniendatei dargestellt, die den Zugriff auf SWF-Dateien gewährt, die aus den Domänen *.example.com, www.friendOfExample.com und 192.0.34.166 stammen:

```
<?xml version="1.0"?>
<cross-domain-policy>
    <allow-access-from domain="*.example.com" />
    <allow-access-from domain="www.friendOfExample.com" />
    <allow-access-from domain="192.0.34.166" />
</cross-domain-policy>
```

Wenn Sie eine IP-Adresse festlegen, erhalten nur die SWF-Dateien Zugriff, die mit der IP-Syntax von der IP-Adresse geladen wurden (z. B. http://65.57.83.12/flashmovie.swf). SWF-Dateien mit Domänennamensyntax wird kein Zugriff gewährt. Flash Player führt keine DNS-Auflösung durch.

Sie können Zugriff auf Dokumente gewähren, die von einer anderen Domäne stammen. Dies wird im folgenden Beispiel gezeigt:

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

Jedes `<allow-access-from>`-Tag verfügt außerdem über das optionale `secure`-Attribut, das standardmäßig den Wert `true` annimmt. Wenn sich die Richtliniendatei auf einem HTTPS-Server befindet und Sie möchten, dass SWF-Dateien auf einem Server ohne HTTPS Daten von dem HTTPS-Server laden können, legen Sie den Wert des Attributs auf `false` fest.

Das Einstellen des `secure`-Attributs auf `false` kann sich jedoch nachteilig auf die von HTTPS gewährleistete Sicherheit auswirken. Insbesondere öffnet das Einstellen dieses Attributs auf `false` sichere Inhalte für Snooping- und Spoofing-Angriffe. Aus diesem Grund wird strikt davon abgeraten, das Attribut `secure` auf `false` einzustellen.

Wenn sich die zu landenden Daten auf einem HTTPS-Server befinden, die SWF-Datei, die die Daten lädt, jedoch auf einem HTTP-Server gespeichert ist, sollten Sie die SWF-Datei auf den HTTPS-Server verschieben. Auf diese Weise können Sie für alle Kopien der sicheren Daten den HTTPS-Schutz aufrechterhalten. Entscheiden Sie sich trotzdem dazu, die ladende SWF-Datei auf einem HTTP-Server beizubehalten, fügen Sie das `secure="false"`-Attribut zu dem `<allow-access-from>`-Tag hinzu. Dies wird im folgenden Code gezeigt:

```
<allow-access-from domain="www.example.com" secure="false" />
```

Ein weiteres Element, das Sie zum gewähren des Zugriffs nutzen können, ist das `allow-http-request-headers-from`-Tag. Dieses Element ermöglicht einem Client, der Inhalte aus einer anderen Berechtigungsdomäne hostet, benutzerdefinierte Header an Ihre Domäne zu senden. während das `<allow-access-from>`-Tag anderen Domänen die Berechtigung gewährt, Daten aus Ihrer Domäne abzurufen, ermöglicht das `allow-http-request-headers-from`-Tag anderen Domänen, Daten in Form von Headern an Ihre Domäne zu übertragen. Im folgenden Beispiel sind alle Domänen berechtigt, den SOAPAction-Header an die aktuelle Domäne zu senden:

```
<cross-domain-policy>
    <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

Wenn die Anweisung `allow-http-request-headers-from` in der Master-Richtliniendatei enthalten ist, wird diese auf alle Verzeichnisse des Hosts angewendet. Andernfalls wird sie nur auf das Verzeichnis der Richtliniendatei, die die Anweisung enthält, und dessen Unterverzeichnisse angewendet.

Vorausladen von Richtliniendateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das laden von Daten von einem Server sowie das Herstellen einer Verbindung zu einem Socket erfolgen asynchron. Flash Player wartet, bis die Richtliniendatei heruntergeladen wurde, bevor der eigentliche Vorgang ausgeführt wird. Das Extrahieren von Pixeldaten aus Bildern oder von Beispieldaten aus Sounds ist jedoch ein synchroner Vorgang. Die Richtliniendatei muss daher geladen werden, bevor Sie die Daten extrahieren. Geben Sie beim Laden der Medien an, dass nach einer Richtliniendatei gesucht werden soll:

- Wenn Sie die `Loader.load()`-Methode verwenden, legen Sie die Eigenschaft `checkPolicyFile` des `context`-Parameters fest, bei dem es sich um ein LoaderContext-Objekt handelt.
- Beim Einbetten eines Bilds in ein Textfeld mit dem `<img>`-Tag stellen Sie das `checkPolicyFile`-Attribut des `<img>`-Tags auf „`true"` ein. Dies wird im Folgenden gezeigt:

  ```
  <img checkPolicyFile = "true" src = "example.jpg">
  ```

- Wenn Sie die `Sound.load()`-Methode verwenden, legen Sie die Eigenschaft `checkPolicyFile` des `context`-Parameters fest, bei dem es sich um ein SoundLoaderContext-Objekt handelt.
- Wenn Sie die NetStream-Klasse verwenden, legen Sie die Eigenschaft `checkPolicyFile` des NetStream-Objekts fest.

Wenn Sie einen dieser Parameter einstellen, prüft Flash Player zunächst auf Richtliniendateien, die bereits für diese Domäne heruntergeladen wurden. Anschließend wird die Richtliniendatei im Standardverzeichnis auf dem Server gesucht. Dabei wird sowohl nach `<allow-access-from>`-Anweisungen als auch nach einer „meta-policy"-Anweisung gesucht. Zum Schluss werden alle ausstehenden Aufrufe der `Security.loadPolicyFile()`-Methode berücksichtigt, um festzustellen, ob sich diese innerhalb des Gültigkeitsbereichs befinden.

Kontrolloptionen für Autoren (Entwickler)

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die zum Erteilen von Sicherheitsrechten verwendete ActionScript-Haupt-API ist die `Security.allowDomain()`-Methode, die SWF-Dateien in den von Ihnen angegebenen Domänen Rechte erteilt. Im folgenden Beispiel wird einer SWF-Datei der Zugriff auf SWF-Dateien erteilt, die sich in der Domäne www.example.com befinden:

```
Security.allowDomain("www.example.com")
```

Diese Methode erteilt Rechte für Folgendes:

- Cross-Scripting zwischen SWF-Dateien (siehe „Cross-Scripting" auf Seite 1138)

- Zugriff auf die Anzeigeliste (siehe „Durchlaufen der Anzeigeliste“ auf Seite 1141)
- Ereigniserkennung (siehe „Sicherheit von Ereignissen“ auf Seite 1141)
- Vollständiger Zugriff auf die Eigenschaften und Methoden des Stage-Objekts (siehe „Sicherheit der Bühne“ auf Seite 1139)

Der Hauptgrund für das Aufrufen der `Security.allowDomain()`-Methode ist das Erteilen von Zugriffsrechten für SWF-Dateien in einer externen Domäne, um die SWF-Datei aufzunehmen, welche die `Security.allowDomain()`-Methode aufruft. Weitere Informationen finden Sie unter „Cross-Scripting“ auf Seite 1138.

Die Angabe einer IP-Adresse als Parameter für die `Security.allowDomain()`-Methode gestattet nicht den Zugriff durch alle Parteien, die von der angegebenen IP-Adresse stammen. Stattdessen erhält hierdurch nur eine Partei Zugriff, die in der URL die angegebene IP-Adresse und nicht den Domänennamen enthält, der dieser IP-Adresse zugeordnet ist. Beispiel: Wenn der Domänenname www.example.com der IP-Adresse 192.0.34.166 zugeordnet ist, gewährt ein Aufruf von `Security.allowDomain("192.0.34.166")` keinen Zugriff auf www.example.com.

Sie können das Platzhalterzeichen „`*`" an die Methode `Security.allowDomain()` übergeben, um den Zugriff von allen Domänen aus zu gestatten. Da das Platzhalterzeichen „*“ SWF-Dateien aus `allen` Domänen Rechte erteilt, ein Cross-Scripting mit der aufrufenden SWF-Datei durchzuführen, müssen Sie es mit Sorgfalt verwenden.

ActionScript enthält eine zweite Berechtigungs-API mit der Bezeichnung `Security.allowInsecureDomain()`. Diese Methode führt das Gleiche aus wie die Methode `Security.allowDomain()`, außer dass sie, wenn sie von einer SWF-Datei aufgerufen wird, die von einer sicheren HTTPS-Verbindung stammt, zusätzlich den Zugriff auf die aufrufende SWF-Datei durch andere SWF-Dateien genehmigt, die von einem nicht sicheren Protokoll, wie z. B. HTTP, stammen. Es stellt jedoch kein gutes Sicherheitsverhalten dar, das Scripting zwischen Dateien von einem sicheren Protokoll (HTTPS) und Dateien von einem nicht sicheren Protokoll (z. B. HTTP) zu gestatten, da sicherer Inhalt verwundbar für Snooping- und Spoofing-Angriffen werden könnte. Derartige Angriffe erfolgen meist nach folgendem Muster: Da die `Security.allowInsecureDomain()`-Methode über von HTTP-Verbindungen stammenden SWF-Dateien Zugriff auf Ihre sicheren HTTPS-Daten gestattet, könnte ein Angreifer zwischen Ihrem HTTP-Server und Ihren Benutzern Ihre HTTP-SWF-Dateien durch eigene ersetzen, die dann auf Ihre HTTPS-Daten zugreifen können.

Wichtig: *Code, der in der AIR-Anwendungs-Sandbox ausgeführt wird, darf die Methoden `allowDomain()` und `allowInsecureDomain()` der Security-Klasse nicht aufrufen.*

Eine weitere wichtige sicherheitsbezogene Methode ist `Security.loadPolicyFile()`, die dazu führt, dass Flash Player an einem nicht dem Standard entsprechenden Speicherort nach einer Richtliniendatei sucht. Weitere Informationen finden Sie unter „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125.

Einschränken von Netzwerk-APIs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Netzwerk-APIs können auf zwei Arten eingeschränkt werden: Um böswillige Angriffe zu verhindern, ist der Zugriff auf allgemein reservierte Ports gesperrt. Dies kann in Ihrem Code nicht außer Kraft gesetzt werden. Verwenden Sie die `allowNetworking`-Einstellung, um den Zugriff einer SWF-Datei auf Netzwerkfunktionen im Hinblick auf andere Ports zu steuern.

Gesperrte Ports

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player und Adobe AIR beschränken den HTTP-Zugriff auf bestimmte Ports, ebenso wie dies in Browsern der Fall ist. HTTP-Anforderungen sind an bestimmten Standardports, die standardmäßig für andere als HTTP-Server verwendet werden, nicht zulässig.

Alle APIs, die auf eine Netzwerk-URL zugreifen, unterliegen diesen Einschränkungen. Die einzige Ausnahme hierzu bilden APIs, die Sockets direkt aufrufen, z. B. `Socket.connect()` und `XMLSocket.connect()`, oder Aufrufe der `Security.loadPolicyFile()`-Methode, in denen eine Socket-Richtliniendatei geladen wird. Socketverbindungen werden durch Verwendung von Socket-Richtliniendateien auf dem Zielserver zugelassen oder verweigert.

In der folgenden Liste sind die ActionScript 3.0-APIs aufgeführt, in denen die Portsperre gilt:

`FileReference.download()`, `FileReference.upload()`, `Loader.load()`, `Loader.loadBytes()`, `navigateToURL()`, `NetConnection.call()`, `NetConnection.connect()`, `NetStream.play()`, `Security.loadPolicyFile()`, `sendToURL()`, `Sound.load()`, `URLLoader.load()`, `URLStream.load()`

Die Portsperre gilt außerdem für den Import freigegebener Bibliotheken, die Verwendung des `<img>`-Tags in Textfeldern und das Laden von SWF-Dateien auf einer HTML-Seite mit den Tags `<object>` und `<embed>`.

Die Portsperre gilt außerdem für die Verwendung des `<img>`-Tags in Textfeldern und das Laden von SWF-Dateien auf einer HTML-Seite mit den Tags `<object>` und `<embed>`.

In der folgenden Liste sind die gesperrten Ports aufgeführt:

HTTP: 20 (ftp data), 21 (ftp control)

HTTP und FTP: 1 (tcpmux), 7 (echo), 9 (discard), 11 (systat), 13 (daytime), 15 (netstat), 17 (qotd), 19 (chargen), 22 (ssh), 23 (telnet), 25 (smtp), 37 (time), 42 (name), 43 (nicname), 53 (domain), 77 (priv-rjs), 79 (finger), 87 (ttylink), 95 (supdup), 101 (hostriame), 102 (iso-tsap), 103 (gppitnp), 104 (acr-nema), 109 (pop2), 110 (pop3), 111 (sunrpc), 113 (auth), 115 (sftp), 117 (uucp-path), 119 (nntp), 123 (ntp), 135 (loc-srv / epmap), 139 (netbios), 143 (imap2), 179 (bgp), 389 (ldap), 465 (smtp+ssl), 512 (print / exec), 513 (login), 514 (shell), 515 (printer), 526 (tempo), 530 (courier), 531 (chat), 532 (netnews), 540 (uucp), 556 (remotefs), 563 (nntp+ssl), 587 (smtp), 601 (syslog), 636 (ldap+ssl), 993 (ldap+ssl), 995 (pop3+ssl), 2049 (nfs), 4045 (lockd), 6000 (x11)

Verwenden des allowNetworking-Parameters

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können den Zugriff einer SWF-Datei auf die Netzwerkfunktionen steuern, indem Sie den Parameter `allowNetworking` in den Tags `<object>` und `<embed>` auf der HTML-Seite mit dem SWF-Inhalt einrichten.

Die zulässigen Werte für `allowNetworking` sind:

- `"all"` (die Standardeinstellung) – Alle Netzwerk-APIs in der SWF-Datei sind zulässig.
- `"internal"` – Die SWF-Datei darf keine der unten in diesem Abschnitt aufgeführten Browsernavigations- oder Browserinteraktions-APIs, aber beliebige andere Netzwerk-APIs aufrufen.
- `"none"` – Die SWF-Datei darf keine der unten in diesem Abschnitt aufgeführten Browsernavigations- oder Browserinteraktions-APIs aufrufen und kann keine der SWF-zu-SWF-Kommunikations-APIs verwenden, die ebenfalls später beschrieben werden.

Der Parameter `allowNetworking` wird vor allem dann verwendet, wenn die SWF-Datei und die zugehörige HTML-Seite von unterschiedlichen Domänen stammen. Die Werte „`internal`" und „`none`" sollten nicht verwendet werden, wenn die geladene SWF-Datei aus der gleichen Domäne wie die zugehörigen HTML-Seiten stammen, da Sie nicht sicherstellen können, dass eine SWF-Datei immer mit der gewünschten HTML-Seite geladen wird. Nicht vertrauenswürdige Parteien könnten in diesem Fall eine SWF-Datei ohne die zugehörige HTML-Seite laden, sodass die `allowNetworking`-Einschränkung nicht mehr wie gewünscht greift.

Der Aufruf einer gesperrten API löst eine SecurityError-Ausnahme aus.

Fügen Sie den Parameter `allowNetworking` hinzu und legen Sie dessen Wert in den Tags `<object>` und `<embed>` auf der HTML-Seite fest, die einen Verweis auf die SWF-Datei enthält. Dies wird im folgenden Beispiel gezeigt:

```
<object classic="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
    Code
base="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,124,
0"
    width="600" height="400" ID="test" align="middle">
<param name="allowNetworking" value="none" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowNetworking="none" bgcolor="#333333"
    width="600" height="400"
    name="test" align="middle" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Eine HTML-Seite kann auch ein Skript verwenden, um SWF-eingebettete Tags zu erzeugen. Sie müssen das Skript ändern, damit die richtigen `allowNetworking`-Einstellungen eingefügt werden. Von Adobe Flash Professional und Adobe Flash Builder generierte HTML-Seiten verwenden die `AC_FL_RunContent()`-Funktion, um Verweise auf SWF-Dateien einzubetten. Fügen Sie dem Skript die Einstellungen für den `allowNetworking`-Parameter hinzu wie im Folgenden:

```
AC_FL_RunContent( ... "allowNetworking", "none", ...)
```

Die folgenden APIs werden gesperrt, wenn `allowNetworking` auf `internal` festgelegt ist:

`navigateToURL()`, `fscommand()`, `ExternalInterface.call()`

Zusätzlich zu den oben aufgeführten APIs werden die folgenden APIs gesperrt, wenn `allowNetworking` auf `none` gesetzt ist:

`sendToURL()`, `FileReference.download()`, `FileReference.upload()`, `Loader.load()`, `LocalConnection.connect()`, `LocalConnection.send()`, `NetConnection.connect()`, `NetStream.play()`, `Security.loadPolicyFile()`, `SharedObject.getLocal()`, `SharedObject.getRemote()`, `Socket.connect()`, `Sound.load()`, `URLLoader.load()`, `URLStream.load()`, `XMLSocket.connect()`

Auch wenn die ausgewählte `allowNetworking`-Einstellung einer SWF-Datei die Verwendung einer Netzwerk-API gestattet, können andere Beschränkungen basierend auf den Einschränkungen der Sicherheits-Sandbox gelten (siehe „Sicherheits-Sandboxen" auf Seite 1117).

Ist `allowNetworking` auf `none` eingestellt, können Sie in einem `<img>`-Tag in der Eigenschaft `htmlText` eines TextField-Objekts nicht auf externe Medien verweisen (es wird eine SecurityError-Ausnahme ausgelöst).

Wenn `allowNetworking` auf `none` festgelegt ist, wird ein Symbol, das aus einer importierten freigegebenen Bibliothek in Flash Professional (nicht ActionScript) hinzugefügt wurde, zur Laufzeit gesperrt.

Sicherheit im Vollbildmodus

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player 9.0.27.0 und höhere Versionen unterstützen den Vollbildmodus, in dem in Flash Player ausgeführter Inhalt den gesamten Bildschirm ausfüllen kann. Zum Aufrufen des Vollbildmodus stellen Sie die Eigenschaft `displayState` der Bühne auf die Konstante `StageDisplayState.FULL_SCREEN` ein. Weitere Informationen finden Sie unter „Verwenden des Vollbildmodus“ auf Seite 178.

Für SWF-Dateien, die in einer Remote-Sandbox ausgeführt werden, gelten einige Sicherheitsaspekte.

Um den Vollbildmodus zu aktivieren, fügen Sie in den Tags `<object>` und `<embed>` auf der HMTL-Seite, die einen Verweis auf die SWF-Datei enthält, den Parameter `allowFullScreen` ein. Legen Sie den Wert des Parameters auf `true` fest (der Standardwert ist `false`). Dies wird im folgenden Beispiel gezeigt:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,
18,0"
    width="600" height="400" id="test" align="middle">
<param name="allowFullScreen" value="true" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowFullScreen="true" bgcolor="#333333"
    width="600" height="400"
    name="test" align="middle" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Eine HTML-Seite kann auch ein Skript verwenden, um SWF-eingebettete Tags zu erzeugen. Sie müssen das Skript so ändern, dass es die richtigen `allowFullScreen`-Einstellungen einfügt. Von Flash Professional und Flash Builder erzeugte HTML-Seiten verwenden die `AC_FL_RunContent()`-Funktion, um Verweise auf SWF-Dateien einzubetten. Sie müssen dann die `allowFullScreen`-Parametereinstellungen wie folgt hinzufügen:

```
AC_FL_RunContent( ... "allowFullScreen", "true", ...)
```

Das ActionScript, das den Vollbildmodus aufruft, kann nur als Reaktion auf ein Maus- oder ein Tastaturereignis aufgerufen werden. Wird es in anderen Situationen aufgerufen, löst Flash Player eine Ausnahme aus.

Wenn der Inhalt im Vollbildmodus angezeigt wird, erscheint eine Meldung, die den Benutzer anweist, wie der Vollbildmodus beendet und wieder der normalen Modus aufgerufen werden kann. Diese Meldung wird einige Sekunden angezeigt und dann ausgeblendet.

Für Inhalte, die in einem Browser ausgeführt werden, wird die Verwendung der Tastatur im Vollbildmodus eingeschränkt. In Flash Player 9 werden nur Tastenkombinationen unterstützt, die die Anwendung in den Modus „normal“ zurückversetzen (z. B. Betätigen der Esc-Taste). Benutzer können keinen Text in die Textfelder eingeben oder durch den Bildschirm navigieren. Ab Flash Player 10 werden bestimmte Tasten für nicht druckbare Zeichen (speziell die Pfeiltasten, die Leertaste und die Tabulatortaste) unterstützt. Die Texteingabe ist jedoch noch immer nicht möglich.

Im eigenständigen Player oder in einer Projektordatei ist der Vollbildmodus immer zulässig. Auch wird die Verwendung der Tastatur (einschließlich Texteingabe) in diesen Umgebungen vollständig unterstützt.

Durch Aufrufen der `displayState`-Eigenschaft eines Stage-Objekts wird für jeden Aufrufer, der sich nicht in derselben Sicherheits-Sandbox wie der Bühneneigentümer (die Haupt-SWF-Datei) befindet, eine Ausnahme ausgelöst. Weitere Informationen finden Sie unter „Sicherheit der Bühne“ auf Seite 1139.

Administratoren können den Vollbildmodus für SWF-Dateien im Browsern deaktivieren, indem sie in der Datei „mms.cfg“ `FullScreenDisable = 1` einstellen. Weitere Informationen finden Sie unter „Kontrolloptionen für Administratoren“ auf Seite 1122.

In einem Browser muss eine SWF-Datei in einer HTML-Datei enthalten sein, damit sie im Vollbildmodus angezeigt werden kann.

Sicherheit im interaktiven Vollbildmodus

Flash Player 11.3 und höher, Adobe AIR 1.0 und höher

Flash Player 11.3 und höhere Versionen unterstützen den interaktiven Vollbildmodus, in dem in Flash Player ausgeführter Inhalt den gesamten Bildschirm ausfüllen *und Texteingaben annehmen* kann. Zum Aufrufen des interaktiven Vollbildmodus stellen Sie die `displayState`-Eigenschaft der Bühne auf die `StageDisplayState.FULL_SCREEN_INTERACTIVE`-Konstante ein. Weitere Informationen finden Sie unter „Verwenden des Vollbildmodus“ auf Seite 178.

Für SWF-Dateien, die in einer Remote-Sandbox ausgeführt werden, gelten einige Sicherheitsaspekte.

Um den interaktiven Vollbildmodus zu aktivieren, fügen Sie in den Tags `<object>` und `<embed>` auf der HMTL-Seite, die einen Verweis auf die SWF-Datei enthält, den Parameter `allowFullScreenInteractive` ein. Legen Sie den Wert des Parameters auf `true` fest (der Standardwert ist `false`). Dies wird im folgenden Beispiel gezeigt:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,
18,0"
    width="600" height="400" id="test" align="middle">
<param name="allowFullScreenInteractive" value="true" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowFullScreen="true" bgcolor="#333333"
    width="600" height="400"
    name="test" align="middle" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Eine HTML-Seite kann auch ein Skript verwenden, um SWF-eingebettete Tags zu erzeugen. Sie müssen das Skript so ändern, dass es die richtigen `allowFullScreenInteractive`-Einstellungen einfügt. Von Flash Professional und Flash Builder erzeugte HTML-Seiten verwenden die `AC_FL_RunContent()`-Funktion, um Verweise auf SWF-Dateien einzubetten. Sie müssen dann die `allowFullScreenInteractive`-Parametereinstellungen wie folgt hinzufügen:

```
AC_FL_RunContent( ... "allowFullScreenInteractive", "true", ...)
```

Das ActionScript, das den interaktiven Vollbildmodus aufruft, kann nur als Reaktion auf ein Maus- oder ein Tastaturereignis aufgerufen werden. Wird es in anderen Situationen aufgerufen, löst Flash Player eine Ausnahme aus.

Es wird eine überlagernde Meldung angezeigt, wenn der Inhalt in den interaktiven Vollbildmodus wechselt. Die Meldung zeigt die Domäne der Vollbildseite, Anweisungen zum Beenden des Vollbildmodus sowie eine **Zulassen**-Schaltfläche an. Die Überlagerung bleibt sichtbar, bis der Benutzer auf **Zulassen** klickt und damit den interaktiven Vollbildmodus bestätigt.

Administratoren können den interaktiven Vollbildmodus für SWF-Dateien im Browser deaktivieren, indem sie in der Datei „mms.cfg“ `FullScreenInteractiveDisable = 1` einstellen. Weitere Informationen finden Sie unter „Kontrolloptionen für Administratoren“ auf Seite 1122.

In einem Browser muss eine SWF-Datei in einer HTML-Datei enthalten sein, damit sie im interaktiven Vollbildmodus angezeigt werden kann.

Laden von Inhalten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player- und AIR-Inhalte können zahlreiche andere Inhaltstypen laden, wie zum Beispiel:

- SWF-Dateien
- Bilder
- Sound
- Video
- HTML-Dateien (nur AIR)
- JavaScript (nur AIR)

Laden von SWF-Dateien und Bildern mit der Loader-Klasse

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Loader-Klasse können Sie SWF-Dateien und Bilder (JPG-, GIF- oder PNG-Dateien) laden. Jede SWF-Datei, außer SWF-Dateien in der „local-with-filesystem“-Sandbox, kann SWF-Dateien und Bilder aus einer beliebigen Netzwerkdomäne laden. Nur SWF-Dateien in lokalen Sandboxen können SWF-Dateien und Bilder aus dem lokalen Dateisystem laden. Dateien in der „local-with-networking“-Sandbox können nur lokale SWF-Dateien laden, die sich in der „local-trusted“- oder der „local-with-networking“-Sandbox befinden. SWF-Dateien in der „local-with-networking“-Sandbox können neben SWF-Dateien auch andere lokale Inhalte (z. B. Bilder) laden, sie können jedoch nicht auf die Daten im geladenen Inhalt zugreifen.

Wenn Sie eine SWF-Datei aus einer nicht vertrauenswürdigen Quelle laden (etwa einer Domäne, die nicht mit der SWF-Stammdatei des Loader-Objekts übereinstimmt), empfiehlt es sich, eine Maske für das Loader-Objekt zu definieren. Dadurch wird verhindert, dass der geladene Inhalt (der dem Loader-Objekt untergeordnet ist) in Bereichen der Bühne gezeichnet wird, die außerhalb dieser Maske liegen. Ein Beispiel hierfür finden Sie im folgenden Code:

```
import flash.display.*;
import flash.net.URLRequest;
var rect:Shape = new Shape();
rect.graphics.beginFill(0xFFFFFF);
rect.graphics.drawRect(0, 0, 100, 100);
addChild(rect);
var ldr:Loader = new Loader();
ldr.mask = rect;
var url:String = "http://www.unknown.example.com/content.swf";
var urlReq:URLRequest = new URLRequest(url);
ldr.load(urlReq);
addChild(ldr);
```

Wenn Sie die `load()`-Methode des Loader-Objekts aufrufen, können Sie einen `context`-Parameter angeben, bei dem es sich um ein LoaderContext-Objekt handelt. Die LoaderContext-Klasse umfasst drei Eigenschaften, mit denen Sie den Kontext definieren können, wie der geladene Inhalt verwendet werden kann:

- `checkPolicyFile`: Verwenden Sie diese Eigenschaft nur beim Laden einer Bilddatei (nicht beim Laden einer SWF-Datei). Dies ist nur dann notwendig, wenn eine Bilddatei aus einer anderen Domäne stammt als die Datei, die das Loader-Objekt enthält. Wenn Sie diese Eigenschaft auf `true` festlegen, sucht der Loader auf dem Ursprungsserver nach einer URL-Richtliniendatei (siehe „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125). Wenn der Server Zugriffsrechte für die Loader-Domäne erteilt, kann ActionScript aus SWF-Dateien in der Loader-Domäne auf Daten im geladenen Bild zugreifen. Anders ausgedrückt, Sie können mit der Eigenschaft `Loader.content` einen Verweis auf das Bitmap-Objekt erhalten, das ein geladenes Bild darstellt; oder Sie verwenden die Methode `BitmapData.draw()` oder `BitmapData.drawWithQuality()`, um auf die Pixel des geladenen Bilds zuzugreifen. Die `drawWithQuality`-Methode ist in Flash Player 11.3 und höher sowie AIR 3.3 und höher verfügbar.
- `securityDomain`: Verwenden Sie diese Eigenschaft nur beim Laden einer SWF-Datei (nicht beim Laden eines Bilds). Dies ist nur dann notwendig, wenn eine SWF-Datei aus einer anderen Domäne stammt als die Datei, die das Loader-Objekt enthält. Für die Eigenschaft `securityDomain` werden derzeit nur zwei Werte unterstützt: `null` (Standardwertdefault) und `SecurityDomain.currentDomain`. Wenn Sie `SecurityDomain.currentDomain` einstellen, muss die geladene SWF-Datei in die Sandbox der ladenden SWF-Datei *importiert* werden. Mit anderen Worten, sie wird so behandelt, als ob sie vom eigenen Server der ladenden SWF-Datei geladen wurde. Dies ist nur dann zulässig, wenn sich eine URL-Richtliniendatei auf dem Server der geladenen SWF-Datei befindet, die den Zugriff auf die Domäne der ladenden SWF-Datei gewährt. Wenn die erforderliche Richtliniendatei gefunden wurde, haben die ladende und die geladene Datei mit Beginn des Ladens gegenseitigen Skriptzugriff, da sich beide Dateien in der gleichen Sandbox befinden. Beachten Sie, dass das Sandbox-Importieren weitestgehend durch das Ausführen eines regulären Ladevorgangs und das Aufrufen der Methode `Security.allowDomain()` durch die geladene SWF-Datei ersetzt werden kann. Das zuletzt genannte Verfahren ist einfacher anzuwenden, da sich die geladene SWF-Datei in ihrer eigenen natürlichen Sandbox befindet und auf Ressourcen auf ihrem eigenen Server zugreifen kann.
- `applicationDomain`: Verwenden Sie diese Eigenschaft nur beim Laden einer SWF-Datei, die in ActionScript 3.0 geschrieben wurde (nicht beim Laden eines Bilds oder einer SWF-Datei, die in ActionScript 1.0 oder 2.0 geschrieben wurde). Beim Laden der Datei können Sie angeben, dass die Datei nicht wie in der Standardeinstellung in einer neuen Anwendungsdomäne (bei der es sich um ein untergeordnetes Element der Anwendungsdomäne der ladenden SWF-Datei handelt), sondern in einer bestimmten Anwendungsdomäne abgelegt werden soll. Beachten Sie, dass Anwendungsdomänen Untereinheiten von Sicherheitsdomänen sind. Aus diesem Grund können Sie nur dann eine Ziel-Anwendungsdomäne angeben, wenn die von Ihnen geladene SWF-Datei aus Ihrer eigenen Sicherheitsdomäne stammt (entweder stammt sie von Ihrem Server oder Sie haben sie mit der Eigenschaft `securityDomain` erfolgreich in Ihre Sicherheitsdomäne importiert). Wenn Sie eine Anwendungsdomäne angegeben haben, die geladene SWF-Datei aber Teil einer anderen Sicherheitsdomäne ist, wird die Domäne, die Sie in `applicationDomain` angegeben haben, ignoriert. Weitere Informationen finden Sie unter „Verwenden von Anwendungsdomänen“ auf Seite 157.

Weitere Informationen finden Sie unter „Festlegen des Ladekontexts“ auf Seite 214.

Eine wichtige Eigenschaft eines Loader-Objekts ist `contentLoaderInfo`, bei der es sich um ein LoaderInfo-Objekt handelt. Im Gegensatz zu den meisten anderen Objekten wird ein LoaderInfo-Objekte von der ladenden SWF-Datei und dem geladenen Inhalt gemeinsam genutzt und beide Parteien können immer darauf zugreifen. Handelt es sich bei dem geladenen Inhalt um eine SWF-Datei, kann sie über die Eigenschaft `DisplayObject.loaderInfo` auf das LoaderInfo-Objekt zugreifen. LoaderInfo-Objekte enthalten Informationen wie den Ladefortschritt, die URLs der ladenden und der geladenen Dateien, die Vertrauensbeziehungen zwischen ladender und geladener Datei und andere Daten. Weitere Informationen finden Sie unter „Überwachen des Ladefortschritts“ auf Seite 212.

Laden von Sound und Video

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Jeglicher Inhalt, mit Ausnahme von Inhalt in der Sandbox „local-with-filesystem“, kann mit den Methoden `Sound.load()`, `NetConnection.connect()` und `NetStream.play()` Sound und Video aus dem Netzwerk laden.

Nur Inhalt in der Sandbox „local-with-filesystem“ und in der AIR-Anwendungs-Sandbox kann Medien aus dem lokalen Dateisystem laden. Nur Inhalt in der Sandbox „local-with-filesystem“, der AIR-Anwendungs-Sandbox und der local-trusted-Sandbox kann auf Daten in diesen geladenen Dateien zugreifen.

Es gelten noch weitere Einschränkungen für den Zugriff auf Daten in geladenen Medien. Weitere Informationen finden Sie unter „Zugriff auf geladene Medien als Daten“ auf Seite 1142.

Laden von SWF-Dateien und Bildern mit dem <img>-Tag in einem Textfeld

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit dem `<img>`-Tag können Sie SWF-Dateien und Bitmaps in ein Textfeld laden. Dies wird im folgenden Beispielcode gezeigt:

```
<img src = 'filename.jpg' id = 'instanceName' >
```

Auf Inhalte, die auf diese Weise geladen wurden, können Sie mit der `getImageReference()`-Methode der TextField-Instanz zugreifen, wie im folgenden Code dargestellt:

```
var loadedObject:DisplayObject = myTextField.getImageReference('instanceName');
```

Beachten Sie jedoch, dass SWF-Dateien und Bilder, die auf diese Weise geladen wurden, einer ihrem Ursprung entsprechenden Sandbox zugewiesen werden.

Wenn Sie eine Bilddatei mit dem `<img>`-Tag in einem Textfeld laden, kann der Zugriff auf die Daten im Bild durch eine URL-Richtliniendatei gestattet sein. Sie können das Vorhandensein einer Richtliniendatei prüfen, indem Sie ein `checkPolicyFile`-Attribut zum `<img>`-Tag hinzufügen, wie im folgenden Code dargestellt:

```
<img src = 'filename.jpg' checkPolicyFile = 'true' id = 'instanceName' >
```

Wenn Sie eine SWF-Datei mit einem `<img>`-Tag in einem Textfeld laden, können Sie den Zugriff auf die Daten dieser SWF-Datei über einen Aufruf der Methode `Security.allowDomain()` gewähren.

Verwenden Sie jedoch ein `<img>`-Tag in einem Textfeld, um eine externe Datei zu laden (anstelle einer Bitmap-Klasse, die in Ihre SWF-Datei eingebettet ist), wird automatisch ein Loader-Objekt als untergeordnetes Element des TextField-Objekts erstellt. Die externe Datei wird in diesen Loader geladen, als ob Sie in ActionScript ein Loader-Objekt zum Laden der Datei verwendet hätten. In diesem Fall gibt die Methode `getImageReference()` den automatisch erstellten Loader zurück. Für den Zugriff auf dieses Loader-Objekt ist keine Sicherheitsprüfung erforderlich, da es sich in der gleichen Sicherheits-Sandbox wie der aufrufende Code befindet.

Wenn Sie jedoch auf die `content`-Eigenschaft des Loader-Objekts verweisen, um auf die geladenen Medien zuzugreifen, kommen Sicherheitsregeln zum Einsatz. Handelt es sich bei dem Inhalt um ein Bild, müssen Sie eine URL-Richtliniendatei implementieren. Handelt es sich bei dem Inhalt um eine SWF-Datei, muss der Code in der SWF-Datei die Methode `allowDomain()` aufrufen.

Adobe AIR

In der Anwendungs-Sandbox werden <img>-Tags in Textfeldern ignoriert, um Phishing-Angriffe zu verhindern. Außerdem darf Code, der in der Anwendungs-Sandbox ausgeführt wird, die `allowDomain()`-Methode der Security-Klasse nicht aufrufen.

Über RTMP-Server bereitgestellte Inhalte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Media Server verwendet zur Bereitstellung von Daten, Audio und Video das Real-Time Media Protocol (RTMP-Protokoll). Sie können diese Medien laden, indem Sie die `connect()`-Methode der NetConnection-Klasse verwenden und dabei eine RTMP-URL als Parameter übergeben. Flash Media Server kann basierend auf der Domäne der anfordernden Datei Verbindungen einschränken und verhindern, dass Inhalte heruntergeladen werden. Weitere Informationen finden Sie in der Onlinedokumentation zu Flash Media Server unter www.adobe.com/go/learn_fms_docs_de.

Damit Grafik- und Sounddaten zur Laufzeit mithilfe der Methoden `BitmapData.draw()`, `BitmapData.drawWithQuality()` und `SoundMixer.computeSpectrum()` aus RTMP-Streams extrahiert werden können, müssen Sie den Zugriff auf dem Server zulassen. Verwenden Sie die serverseitigen ActionScript-Eigenschaften `Client.videoSampleAccess` und `Client.audioSampleAccess`, um den Zugriff auf bestimmte Verzeichnisse auf dem Flash Media Server zu gewähren. Weitere Informationen finden Sie im Handbuch Server-Side ActionScript Language Reference. (Die `drawWithQuality`-Methode ist in Flash Player 11.3 und höher sowie AIR 3.3 und höher verfügbar.)

Cross-Scripting

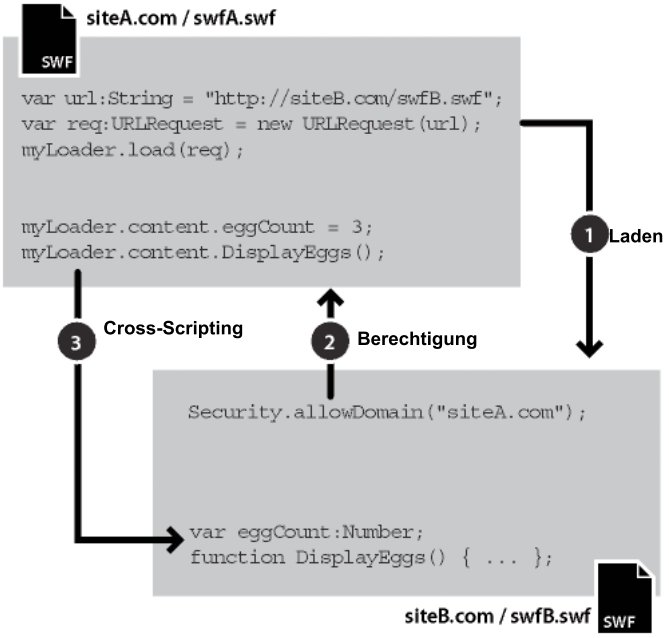
Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn zwei mit ActionScript 3.0 geschriebene SWF-Dateien oder zwei in AIR ausgeführte HTML-Dateien von derselben Domäne bereitgestellt werden (beispielsweise lautet die URL für eine SWF-Datei „http://www.example.com/swfA.swf“ und für die andere „http://www.example.com/swfB.swf“), kann der in einer Datei definierte Code die Variablen, Objekte, Eigenschaften, Methoden usw. der anderen Datei untersuchen und ändern und umgekehrt. Dies wird als *Cross-Scripting* bezeichnet.

Werden die beiden Dateien von verschiedenen Domänen bereitgestellt (beispielsweise „http://siteA.com/swfA.swf“ und „http://siteB.com/swfB.swf“), ist es standardmäßig in Flash Player und AIR nicht möglich, dass „swfA.swf“ eine Skripterstellung für „swfB.swf“ durchführt und umgekehrt. Eine SWF-Datei erteilt SWF-Dateien von anderen Domänen Berechtigungen durch Aufrufen von `Security.allowDomain()`. Durch Aufrufen von `Security.allowDomain("siteA.com")` gewährt „swfB.swf“ SWF-Dateien von „siteA.com“ eine Berechtigung für den Skriptzugriff.

Cross-Scripting zwischen AVM1 SWF-Dateien und AVM2 SWF-Dateien wird nicht unterstützt. Eine AVM1 SWF-Datei ist eine in ActionScript 1.0 oder 2.0 geschriebene SWF-Datei. (AVM1 und AVM2 beziehen sich auf ActionScript Virtual Machine.) Sie können jedoch die LocalConnection-Klasse für den Datenaustausch zwischen AVM1 und AVM2 verwenden.

In domänenübergreifenden Situationen ist es wichtig, die beiden Seiten klar zu trennen. In der vorliegenden Erläuterung soll die Seite, die das Cross-Scripting durchführt, als *zugreifende Seite* (normalerweise die zugreifende SWF-Datei) und die andere Seite als die *Seite, auf die zugegriffen wird* (normalerweise die SWF-Datei, auf die zugegriffen wird) bezeichnet werden. Wenn „siteA.swf" einen Skriptzugriff auf „siteB.swf" durchführt, handelt es sich bei „siteA.swf" um die zugreifende Seite und bei „siteB.swf" um die Seite, auf die zugegriffen wird. Dies wird in der folgenden Abbildung verdeutlicht:



Domänenübergreifende Zugriffsrechte, die mit `Security.allowDomain()` erstellt wurden, sind asymmetrisch. Im vorangegangenen Beispiel kann „siteA.swf" einen Skriptzugriff auf „siteB.swf" durchführen, „siteB.swf" jedoch nicht auf „siteA.swf", da „siteA.swf" nicht die Methode `Security.allowDomain()` aufgerufen hat, um SWF-Dateien auf „siteB.com" die Berechtigung zum Skriptzugriff zu erteilen. Symmetrische Zugriffsrechte werden eingerichtet, indem beide SWF-Dateien die Methode `Security.allowDomain()` aufrufen.

Flash Player schützt SWF-Dateien nicht nur vor Cross-Domain-Scripting durch andere SWF-Dateien, sondern auch vor Cross-Domain-Scripting durch HTML-Dateien. HTML-zu-SWF-Scripting kann bei Rückrufen auftreten, die mit der `ExternalInterface.addCallback()`-Methode hergestellt wurden. Bei domänenübergreifendem HTML-zu-SWF-Skriptzugriff muss die SWF-Datei, auf die zugegriffen wird, `Security.allowDomain()` aufrufen, so als ob es sich bei der zugreifenden Seite um eine SWF-Datei handelt, andernfalls schlägt der Vorgang fehl. Weitere Informationen finden Sie unter „Kontrolloptionen für Autoren (Entwickler)" auf Seite 1129.

Darüber hinaus stellt Flash Player Sicherheitskontrollen für das SWF-zu-HTML-Scripting zur Verfügung. Weitere Informationen finden Sie unter „Steuern des externen URL-Zugriffs" auf Seite 1149.

Sicherheit der Bühne

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Einige Eigenschaften und Methoden des Stage-Objekts stehen allen Sprites oder Movieclips in der Anzeigeliste zur Verfügung.

Das Stage-Objekt verfügt jedoch über einen Eigentümer: die erste geladene SWF-Datei. Standardmäßig stehen die folgenden Eigenschaften und Methoden des Stage-Objekts nur den SWF-Dateien in der gleichen Sicherheits-Sandbox wie der des Bühneneigentümers zur Verfügung:

Eigenschaften	Methoden
`align`	`addChild()`
displayState	`addChildAt()`
frameRate	`addEventListener()`
`height`	`dispatchEvent()`
`mouseChildren`	`hasEventListener()`
`numChildren`	`setChildIndex()`
`quality`	`willTrigger()`
`scaleMode`	
`showDefaultContextMenu`	
`stageFocusRect`	
`stageHeight`	
`stageWidth`	
`tabChildren`	
`textSnapshot`	
`width`	

Damit eine SWF-Datei in einer anderen Sandbox als der des Bühneneigentümers auf diese Eigenschaften und Methoden zugreifen kann, muss die SWF-Datei des Bühneneigentümers die Methode `Security.allowDomain()` aufrufen, um der Domäne in der externen Sandbox Zugriffsrechte zu erteilen. Weitere Informationen finden Sie unter „Kontrolloptionen für Autoren (Entwickler)“ auf Seite 1129.

Die `frameRate`-Eigenschaft ist ein Sonderfall, da die `frameRate`-Eigenschaft von jeder SWF-Datei gelesen werden kann. Jedoch können nur die SWF-Dateien in der Sicherheits-Sandbox des Bühneneigentümers (oder SWF-Dateien, denen durch einen Aufruf der Methode `Security.allowDomain()` entsprechende Zugriffsrechte erteilt wurden) diese Eigenschaft ändern.

Darüber hinaus gibt es einige Einschränkungen bei den Methoden `removeChildAt()` und `swapChildrenAt()` des Stage-Objekts, die sich von den anderen Einschränkungen unterscheiden. Code zum Aufrufen dieser Methoden muss sich in der gleichen Domäne wie der Eigentümer der betroffenen untergeordneten Objekte befinden (und nicht in der gleichen Domäne wie der Bühneneigentümer), oder die untergeordneten Objekte können die Methode `Security.allowDomain()` aufrufen.

Durchlaufen der Anzeigeliste

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Die Fähigkeit einer SWF-Datei, auf Anzeigeobjekte zuzugreifen, die aus anderen Sandboxen geladen werden, ist eingeschränkt. Damit eine SWF-Datei auf ein Anzeigeobjekt zugreifen kann, das von einer SWF-Datei in einer anderen Sandbox erstellt wurde, muss die SWF-Datei, auf die zugegriffen wird, die Methode `Security.allowDomain()` aufrufen, um den Zugriff durch die Domäne der zugreifenden SWF-Datei zu gestatten. Weitere Informationen finden Sie unter „Kontrolloptionen für Autoren (Entwickler)“ auf Seite 1129.

Um auf ein Bitmap-Objekt zuzugreifen, das von einem Loader-Objekt geladen wurde, muss eine URL-Richtliniendatei auf dem Ursprungsserver der Bilddatei vorhanden sein und diese Richtliniendatei muss der Domäne der SWF-Datei, die auf das Bitmap-Objekt zuzugreifen versucht, Zugriffsrechte erteilen (siehe „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125).

Das LoaderInfo-Objekt, das einer geladenen Datei entspricht (und dem Loader-Objekt), enthält die folgenden drei Eigenschaften, mit denen die Beziehung zwischen dem geladenen Objekt und dem Loader-Objekt definiert wird: `childAllowsParent`, `parentAllowsChild` und `sameDomain`.

Sicherheit von Ereignissen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für den Zugriff auf anzeigelistenbezogene Ereignisse bestehen Einschränkungen, die auf der Sandbox des Anzeigeobjekts basieren, welches das Ereignis auslöst. Ein Ereignis in der Anzeigeliste durchläuft eine Aufstiegs- und eine Empfangsphase (siehe Beschreibung in „Verarbeiten von Ereignissen“ auf Seite 133). Während der Aufstiegs- und Empfangsphasen migriert ein Ereignis vom Quell-Anzeigeobjekt über die übergeordneten Anzeigeobjekte in der Anzeigeliste. Befindet sich ein übergeordnetes Objekt in einer anderen Sicherheits-Sandbox als das Quell-Anzeigeobjekt, stoppen Empfangs- und Aufstiegsphase unter diesem übergeordneten Objekt, es sei denn, es besteht ein gegenseitiges Vertrauensverhältnis zwischen dem Eigentümer des übergeordneten Objekts und dem Eigentümer des Quellobjekts. Dieses gegenseitige Vertrauensverhältnis wird folgendermaßen erreicht:

1. Die SWF-Datei, die Eigentümer des übergeordneten Objekts ist, muss die Methode `Security.allowDomain()` aufrufen, um der Domäne der SWF-Datei zu vertrauen, die Eigentümer des Quellobjekts ist.
2. Die SWF-Datei, die Eigentümer des Quellobjekts ist, muss die Methode `Security.allowDomain()` aufrufen, um der Domäne der SWF-Datei zu vertrauen, die Eigentümer des übergeordneten Objekts ist.

Das LoaderInfo-Objekt, das einer geladenen Datei entspricht (und dem Loader-Objekt), enthält die folgenden zwei Eigenschaften, mit denen die Beziehung zwischen dem geladenen Objekt und dem Loader-Objekt definiert wird: `childAllowsParent` und `parentAllowsChild`.

Für Ereignisse, die von anderen Objekten als Anzeigeobjekten ausgelöst werden, gibt es keine Sicherheitsprüfungen oder sicherheitsbezogene Auswirkungen.

Zugriff auf geladene Medien als Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zum Zugriff auf geladene Daten verwenden Sie die Methoden `BitmapData.draw()`, `BitmapData.drawWithQuality()` und `SoundMixer.computeSpectrum()`. Standardmäßig ist es nicht möglich, Pixeldaten oder Audiodaten von Grafik- oder Audioobjekten abzurufen, die von Medien, die in einer anderen Sandbox geladen sind, dargestellt oder abgespielt werden. Sie können jedoch folgende Methoden verwenden, um den sandboxübergreifenden Zugriff auf solche Daten zu gewähren:

- Rufen Sie in dem Inhalt, in dem die gewünschten Daten dargestellt oder abgespielt werden, die `Security.allowDomain()`-Methode auf, um den Datenzugriff auf Inhalt in anderen Domänen zu gewähren.
- Bei einem geladenen Bild, Sound oder Video fügen Sie dem Server der geladenen Datei eine URL-Richtliniendatei hinzu. Diese Richtliniendatei muss Zugriff auf die Domäne der SWF-Datei gewähren, die versucht, die Methoden `BitmapData.draw()`, `BitmapData.drawWithQuality()` oder `SoundMixer.computeSpectrum()` aufzurufen, um Daten aus der Datei zu extrahieren. Die `drawWithQuality`-Methode ist in Flash Player 11.3 und höher sowie AIR 3.3 und höher verfügbar.

In den folgenden Abschnitten finden Sie ausführliche Informationen über den Zugriff auf Bitmap-, Sound- und Videodaten.

Zugriff auf Bitmap-Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Methode `draw()` oder `drawWithQuality()` (Flash Player 11.3; AIR 3.3) eines BitmapData-Objekts können Sie die aktuell angezeigten Pixel eines beliebigen Anzeigeobjekts in ein BitmapData-Objekt zeichnen. Dabei kann es sich um die Pixel eines MovieClip-Objekts, eines Bitmap-Objekts oder eines Anzeigeobjekts handeln. Die folgenden Bedingungen müssen zutreffen, damit diese Methoden Pixel in das BitmapData-Objekt schreiben:

- Handelt es sich bei dem Quellobjekt nicht um eine geladene Bitmap, müssen das Quellobjekt und (bei einem Sprite- oder MovieClip-Objekt) alle untergeordneten Objekte aus der gleichen Domäne wie das Objekt stammen, das die draw-Methode aufruft, oder sie müssen sich in einer SWF-Datei befinden, die dem aufrufenden Objekt durch Aufrufen der Methode `Security.allowDomain()` zugänglich ist.
- Handelt es sich bei dem Quellobjekt um eine geladene Bitmap, muss das Quellobjekt aus der gleichen Domäne wie das Objekt stammen, das die draw-Methode aufruft, oder der Quellserver muss eine URL-Richtliniendatei enthalten, die der aufrufenden Domäne Zugriffsrechte erteilt.

Wenn diese Bedingungen nicht erfüllt werden, wird eine SecurityError-Ausnahme ausgelöst.

Wenn Sie das Bild mit der `load()`-Methode der Loader-Klasse laden, können Sie einen `context`-Parameter angeben, bei dem es sich um ein LoaderContext-Objekt handelt. Wenn Sie die `checkPolicyFile`-Eigenschaft des LoaderContext-Objekts auf `true` festlegen, wird in Flash Player auf dem Server, von dem das Bild geladen wird, eine URL-Richtliniendatei gesucht. Wenn eine Richtliniendatei vorhanden und die Domäne der ladenden SWF-Datei darin enthalten ist, kann die Datei auf die Daten im Bitmap-Objekt zugreifen. Andernfalls wird der Zugriff verweigert.

Sie können auch eine `checkPolicyFile`-Eigenschaft in einem Bild angeben, das über ein `<img>`-Tag in einem Textfeld geladen wird. Weitere Informationen finden Sie unter „Laden von SWF-Dateien und Bildern mit dem <img>-Tag in einem Textfeld“ auf Seite 1137.

Zugriff auf Sounddaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für die folgenden soundbezogenen ActionScript 3.0-APIs gelten Sicherheitseinschränkungen:

- `SoundMixer.computeSpectrum()`-Methode – Für Code, der in der gleichen Sicherheits-Sandbox wie die Sounddatei ausgeführt wird, immer zulässig. Bei Code, der in anderen Sandboxen ausgeführt wird, werden Sicherheitsprüfungen durchgeführt.
- `SoundMixer.stopAll()`-Methode – Für Code, der in der gleichen Sicherheits-Sandbox wie die Sounddatei ausgeführt wird, immer zulässig. Bei Dateien in anderen Sandboxen werden Sicherheitsprüfungen durchgeführt.
- `id3`-Eigenschaft der Sound-Klasse – Für SWF-Dateien, die sich in der gleichen Sicherheits-Sandbox wie die Sounddatei befinden, immer zulässig. Bei Code, der in anderen Sandboxen ausgeführt wird, werden Sicherheitsprüfungen durchgeführt.

Jedem Sound sind zwei Arten von Sandboxen zugeordnet: eine Inhalt-Sandbox und eine Eigentümer-Sandbox.

- Die Ursprungsdomäne des Sounds legt die Inhalt-Sandbox fest und diese bestimmt, ob Daten aus dem Sound über die `id3`-Eigenschaft des Sounds und die `SoundMixer.computeSpectrum()`-Methode extrahiert werden.
- Das Objekt, das die Wiedergabe des Sounds gestartet hat, legt die Eigentümer-Sandbox fest und diese bestimmt, ob der Sound mit der `SoundMixer.stopAll()`-Methode gestoppt werden kann.

Wenn Sie den Sound mit der `load()`-Methode der Sound-Klasse laden, können Sie einen `context`-Parameter angeben, bei dem es sich um ein SoundLoaderContext-Objekt handelt. Wenn Sie die `checkPolicyFile`-Eigenschaft des SoundLoaderContext-Objekts auf `true` setzen, sucht die Laufzeit auf dem Server, von dem der Sound geladen wird, eine URL-Richtliniendatei. Wenn eine Richtliniendatei vorhanden und die Domäne des ladenden Codes darin enthalten ist, kann der Code auf die `id`-Eigenschaft des Sound-Objekts zugreifen. Andernfalls wird der Zugriff verweigert. Darüber hinaus kann auch das Festlegen der `checkPolicyFile`-Eigenschaft die `SoundMixer.computeSpectrum()`-Methode für geladene Sounds aktivieren.

Mit der Methode `SoundMixer.areSoundsInaccessible()` können Sie ermitteln, ob ein Aufruf der `SoundMixer.stopAll()`-Methode alle Sounds stoppen würde, da die Sandbox eines oder mehrerer Sound-Eigentümer für das aufrufende Objekt nicht zugänglich ist.

Das Aufrufen der `SoundMixer.stopAll()`-Methode stoppt Sounds, deren Eigentümer-Sandbox die Gleiche wie die des aufrufenden Objekts von `stopAll()` ist. Sie stoppt auch die Sounds, deren Wiedergabe von SWF-Dateien gestartet wurde, welche die `Security.allowDomain()`-Methode aufgerufen haben, um Zugriff für die Domäne der SWF-Datei zu gewähren, welche die `stopAll()`-Methode aufruft. Alle anderen Sounds werden nicht gestoppt. Das Vorhandensein dieser Sounds kann durch Aufrufen der `SoundMixer.areSoundsInaccessible()`-Methode ermittelt werden.

Das Aufrufen der `computeSpectrum()`-Methode erfordert, dass jeder wiedergegebene Sound entweder aus der gleichen Sandbox wie das Objekt stammt, das die Methode aufgerufen hat, oder von einer Quelle, der ein Zugriffsrecht für die Sandbox des aufrufenden Objekts erteilt wurde. Andernfalls wird eine SecurityError-Ausnahme ausgelöst. Bei Sounds, die aus den eingebetteten Sounds einer Bibliothek in einer SWF-Datei geladen werden, wird das Zugriffsrecht mit dem Aufruf der `Security.allowDomain()`-Methode in der geladenen SWF-Datei erteilt. Bei Sounds, die aus anderen Quellen als SWF-Dateien geladen werden (aus geladenen MP3-Dateien oder aus Videodateien), erteilt eine URL-Richtliniendatei auf dem Quellserver Zugriffsberechtigungen auf die Daten in den geladenen Medien.

Weitere Informationen finden Sie unter „Kontrolloptionen für Autoren (Entwickler)“ auf Seite 1129 und „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125.

Zum Zugriff auf Sounddaten von RTMP-Streams müssen Sie den Zugriff auf dem Server gewähren. Verwenden Sie die serverseitige ActionScript-Eigenschaft `Client.audioSampleAccess`, um den Zugriff auf bestimmte Verzeichnisse auf dem Flash Media Server zu gewähren. Weitere Informationen finden Sie im Handbuch Server-Side ActionScript Language Reference.

Zugriff auf Videodaten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der `BitmapData.draw()`- oder `BitmapData.drawWithQuality()`-Methode können Sie Pixeldaten des aktuellen Bilds eines Videos erfassen. (Die `drawWithQuality`-Methode ist in Flash Player 11.3 und höher sowie AIR 3.3 und höher verfügbar.)

Es gibt zwei verschiedene Arten von Video:

- Über RTMP von Flash Media Server gestreamtes Video
- Progressives Video, das aus einer FLV- oder F4V-Datei geladen wird

Damit die draw-Methoden zum Extrahieren von Laufzeitgrafiken aus RTMP-Streams verwendet werden können, müssen Sie den Zugriff auf dem Server gewähren. Verwenden Sie die serverseitige ActionScript-Eigenschaft `Client.videoSampleAccess`, um den Zugriff auf bestimmte Verzeichnisse auf dem Flash Media Server zu gewähren. Weitere Informationen finden Sie im Handbuch Server-Side ActionScript Language Reference.

Wenn Sie eine draw-Methode mit progressivem Video als `source`-Parameter aufrufen, muss sich das Objekt, das die Methode aufruft, entweder in der gleichen Sandbox wie die FLV-Datei befinden, oder auf dem Server der FLV-Datei ist eine Richtliniendatei gespeichert, in der Zugriffsrechte für die Domäne der aufrufenden SWF-Datei erteilt werden. Durch Festlegen der `checkPolicyFile`-Eigenschaft des NetStream-Objekts auf `true` können Sie das Herunterladen der Richtliniendatei anfordern.

Laden von Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Flash Player- und AIR-Inhalte können Daten mit Servern austauschen. Das Laden von Daten ist ein anderer Vorgang als das Laden von Medien, da die geladenen Informationen als Programmobjekte und nicht als Medien angezeigt werden. Im Allgemeinen kann Inhalt Daten aus der Domäne laden, aus der der Inhalt stammt. In der Regel erfordert Inhalt jedoch Richtliniendateien, um Daten aus anderen Domänen zu laden; siehe „Kontrolloptionen für Websites (Richtliniendateien)“ auf Seite 1125.

Hinweis: *Inhalt, der in der AIR-Anwendungs-Sandbox ausgeführt wird, wird nie von einer Remote-Domäne bereitgestellt (es sei denn, der Entwickler importiert den Remote-Inhalt explizit in die Anwendungs-Sandbox); deshalb ist dieser Inhalt immun gegenüber den Angriffen, die domänenübergreifende Richtlinien verhindern. Richtliniendateien verhindern nicht, dass AIR-Inhalt in der Anwendungs-Sandbox Daten laden kann. Für AIR-Inhalt in anderen Sandboxen gelten jedoch die hier beschriebenen Einschränkungen.*

Verwenden von URLLoader und URLStream

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Sie können Daten laden, z. B. eine XML-Datei oder eine Textdatei. Die `load()`-Methoden der URLLoader- und der URLStream-Klasse unterliegen den Zugriffsberechtigungen in einer URL-Richtliniendatei.

Wenn Sie die `load()`-Methode zum Laden von Inhalten aus einer anderen Domäne verwenden als der, in der sich der diese Methode aufrufende Code befindet, sucht die Laufzeit auf dem Server mit den geladenen Elementen nach einer URL-Richtliniendatei. Wenn eine Richtliniendatei vorhanden ist und sie den Zugriff auf die Domäne des ladenden Inhalts erteilt, können Sie die Daten laden.

Herstellen einer Verbindung mit Sockets

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Standardmäßig sucht die Laufzeit nach einer Socket-Richtliniendatei, die an Port 843 bereitgestellt wird. Wie bei URL-Richtliniendateien wird diese Datei als *Masterrichtliniendatei* bezeichnet.

Als Richtliniendateien in Flash Player 6 eingeführt wurden, gab es noch keine Unterstützung für Socket-Richtliniendateien. Verbindungen mit Socketservern wurden von einer Richtliniendatei am Standardspeicherort auf einem HTTP-Server an Port 80 auf dem gleichen Host wie der Socketserver autorisiert. Diese Funktion wird in Flash Player 9 noch unterstützt, nicht jedoch in Flash Player 10. In Flash Player 10 können ausschließlich Socket-Richtliniendateien Socketverbindungen autorisieren.

Socket-Richtliniendateien unterstützen ähnlich wie URL-Richtliniendateien eine „meta-policy“-Anweisung, die angibt, welche Ports Richtliniendateien bereitstellen können. Die standardmäßige „meta-policy“-Anweisung für Socket-Richtliniendateien ist jedoch „all“, nicht „master-only“. Flash Player geht somit davon aus, dass jedes Socket auf dem Host eine Socket-Richtliniendatei bereitstellen kann, sofern die Master-Richtliniendatei keine strikteren Einstellungen festlegt.

Der Zugriff auf Socket- und XML-Socketverbindungen ist standardmäßig deaktiviert, selbst wenn sich das Socket, zu dem Sie eine Verbindung herstellen möchten, in der gleichen Domäne wie die SWF-Datei befindet. Um den Zugriff auf Socketebene zuzulassen, müssen Sie eine Socket-Richtliniendatei in einem der folgenden Speicherorte bereitstellen:

- Port 843 (Speicherort der Master-Richtliniendatei)
- Der gleiche Port wie die Socket-Hauptverbindung
- Ein anderer Port als die Socket-Hauptverbindung

Flash Player sucht standardmäßig an Port 843 sowie an dem Port der Socket-Hauptverbindung nach einer Socket-Richtliniendatei. Wenn Sie eine Socket-Richtliniendatei an einem anderen Port bereitstellen möchten, muss die SWF-Datei die Methode `Security.loadPolicyFile()` aufrufen.

Eine Socket-Richtliniendatei weist die gleiche Syntax wie eine URL-Richtliniendatei auf, sie gibt jedoch zusätzlich die Ports an, für die Zugriff gewährt wird. Wenn eine Socket-Richtliniendatei von einer Portnummer unter 1024 bereitgestellt wird, kann sie Zugriff auf beliebige Ports gewähren. Stammt sie dagegen von Port 1024 oder höher, kann sie nur Zugriff auf Port 1024 und höher gewähren. Die zulässigen Ports werden in einem `to-ports`-Attribut im `<allow-access-from>`-Tag angegeben. Einstellige Portnummern, Portbereiche und Platzhalterzeichen sind zulässig.

Im Folgenden finden Sie ein Beispiel für eine Socket-Richtliniendatei:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<!-- Policy file for xmlsocket://socks.mysite.com -->
<cross-domain-policy>
    <allow-access-from domain="*" to-ports="507" />
    <allow-access-from domain="*.example.com" to-ports="507,516" />
    <allow-access-from domain="*.example.org" to-ports="516-523" />
    <allow-access-from domain="adobe.com" to-ports="507,516-523" />
    <allow-access-from domain="192.0.34.166" to-ports="*" />
</cross-domain-policy>
```

Um eine Socket-Richtliniendatei an Port 843 oder dem gleichen Port wie die Socket-Hauptverbindung abzurufen, rufen Sie die Methode `Socket.connect()` oder `XMLSocket.connect()` auf. Flash Player sucht zunächst eine Master-Richtliniendatei an Port 843. Wenn eine Master-Richtliniendatei gefunden wird, überprüft Flash Player, ob die Datei eine „meta-policy"-Anweisung enthält, die Socket-Richtliniendateien am Zielport untersagt. Ist der Zugriff zulässig, sucht Flash Player in der Master-Richtliniendatei zunächst nach der zugehörigen `allow-access-from`-Anweisung. Wird keine Master-Richtliniendatei gefunden, sucht Flash Player am gleichen Port wie die Socket-Hauptverbindung nach einer Socket-Richtliniendatei.

Um eine Socket-Richtliniendatei aus einem anderen Speicherort abzurufen, müssen Sie zunächst die Methode `Security.loadPolicyFile()` mit der speziellen `"xmlsocket"`-Syntax aufrufen. Dies wird im folgenden Beispiel gezeigt:

```
Security.loadPolicyFile("xmlsocket://server.com:2525");
```

Rufen Sie die Methode `Security.loadPolicyFile()` auf, bevor Sie die `Socket.connect()`- oder `XMLSocket.connect()`-Methode aufrufen. In diesem Fall wartet Flash Player, bis Ihre Richtliniendateianforderung erfüllt wurde. Erst dann entscheidet es, ob Ihre Hauptverbindung zugelassen wird. Wenn die Master-Richtliniendatei jedoch festlegt, dass das Zielverzeichnis keine Richtliniendateien bereitstellen kann, hat der Aufruf der Methode `loadPolicyFile()` keine Auswirkung, selbst wenn in diesem Verzeichnis eine Richtliniendatei vorhanden ist.

Wenn Sie einen Socketserver implementieren und eine Socket-Richtliniendatei bereitstellen müssen, entscheiden Sie, ob die Richtliniendatei an dem gleichen Port bereitgestellt werden soll, der Hauptverbindungen akzeptiert, oder ob Sie einen anderen Port verwenden möchten. In beiden Fällen muss der Server auf die erste Übertragung vom Client warten, bevor er eine Antwort sendet.

Wenn Flash Player eine Richtliniendatei anfordert, überträgt es nach dem Herstellen der Verbindung immer zuerst den folgenden String:

```
<policy-file-request/>
```

Erst nachdem der Server diesen String empfangen hat, kann er die Richtliniendatei senden. Die Anforderung von Flash Player wird immer durch ein Nullbyte beendet und die Antwort des Servers muss ebenfalls auf ein Nullbyte enden.

Erwarten Sie nicht, dass die gleiche Verbindung für die Richtliniendateianforderung und die Hauptverbindung verwendet wird. Sie müssen die Verbindung nach Übertragung der Richtliniendatei schließen. Andernfalls schließt Flash Player die Verbindung für die Richtliniendatei, bevor die Hauptverbindung eingerichtet wird.

Schützen von Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Um Daten bei der Übertragung über das Internet vor Abhör- und Änderungsversuchen zu schützen, können Sie TLS (Transport Layer Security) oder SSL (Secure Socket Layer) auf dem Server verwenden, von dem die Daten stammen. Dann können Sie über das HTTPS-Protokoll eine Verbindung mit dem Server herstellen.

In Anwendungen, die für AIR 2 oder höher erstellt werden, lässt sich auch die TCP-Socket-Kommunikation schützen. Mithilfe der SecureSocket-Klasse können Sie eine Socketverbindung mit einem Socket-Server einleiten, der TLS-Version 1 oder SSL-Version 4 verwendet.

Senden von Daten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Das Senden von Daten tritt auf, wenn der Code Daten an einen Server oder an eine Ressource sendet. Das Senden von Daten ist für Inhalt aus einer Netzwerkdomäne immer zulässig. Eine lokale SWF-Datei kann nur dann Daten an Netzwerkadressen senden, wenn sie sich in der „local-trusted"-Sandbox, der „local-with-networking"-Sandbox oder der AIR-Anwendungs-Sandbox befindet. Weitere Informationen finden Sie unter „Lokale Sandboxen" auf Seite 1117.

Zum Senden von Daten an eine URL können Sie die Funktion `flash.net.sendToURL()` verwenden. Andere Methoden senden ebenfalls Anforderungen an URLs. Hierzu gehören ladende Methoden wie `Loader.load()` und `Sound.load()` und datenladende Methoden wie `URLLoader.load()` und `URLStream.load()`.

Hoch- und Herunterladen von Dateien

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der Methode `FileReference.upload()` starten Sie das Hochladen einer ausgewählten Datei auf einen Remote-Server. Bevor Sie die `FileReference.upload()`-Methode aufrufen können, müssen Sie die `FileReference.browse()`- oder die `FileReferenceList.browse()`-Methode aufrufen.

Der Code, der die Methode `FileReference.browse()` oder `FileReferenceList.browse()` initiiert, kann nur als Reaktion auf ein Maus- oder ein Tastaturereignis aufgerufen werden. Wird er in anderen Situationen aufgerufen, lösen Flash Player 10 und neuere Versionen eine Ausnahme aus. Ein vom Benutzer initiiertes Ereignis ist jedoch nicht erforderlich, um diese Methoden von der AIR-Anwendungs-Sandbox aus aufzurufen.

Das Aufrufen der Methode `FileReference.download()` öffnet ein Dialogfeld, in dem der Benutzer eine Datei zum Herunterladen von einem Remote-Server auswählen kann.

Hinweis: *Wenn der Server eine Benutzerauthentifizierung erfordert, kann der Benutzer nur bei SWF-Dateien, die in einem Browser — also mit einem Browser-Plug-In oder einer ActiveX-Steuerung — ausgeführt werden, in einem Dialogfeld zur Eingabe eines Benutzernamens und eines Kennworts zur Authentifizierung aufgefordert werden. Dies gilt darüber hinaus nur für Download-Vorgänge. Flash Player gestattet nicht das Hochladen auf einen Server, der eine Benutzerauthentifizierung erfordert.*

Das Hoch- und Herunterladen wird nicht gestattet, wenn sich die aufrufende SWF-Datei in der „local-with-filesystem"-Sandbox befindet.

In der Standardeinstellung kann eine SWF-Datei nur das Hoch- und Herunterladen vom bzw. auf den eigenen Server initiieren. Eine SWF-Datei kann auch auf einen anderen Server hoch- bzw. davon herunterladen, wenn dieser Server eine Richtliniendatei bereitstellt, in der Zugriffsberechtigungen für die Domäne der aufrufenden SWF-Datei erteilt werden.

Laden von eingebetteten Inhalten aus SWF-Dateien, die in eine Sicherheitsdomäne importiert wurden

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Beim Laden einer SWF-Datei können Sie den Parameter `context` der `load()`-Methode des Loader-Objekts einstellen, das zum Laden der Datei verwendet wird. Dieser Parameter arbeitet mit einem LoaderContext-Objekt. Wenn Sie die `securityDomain`-Eigenschaft dieses LoaderContext-Objekts auf `Security.currentDomain` einstellen, sucht Flash Player auf dem Server der geladenen SWF-Datei nach einer URL-Richtliniendatei. Wenn eine Richtliniendatei vorhanden ist und sie Zugriff auf die Domäne der ladenden SWF-Datei erteilt, können Sie die SWF-Datei als importierte Medien laden. Auf diese Weise erhält die ladende Datei Zugriff auf Objekte in der Bibliothek der SWF-Datei.

Alternativ kann eine SWF-Datei auf die Klassen in geladenen SWF-Dateien aus einer anderen Sicherheits-Sandbox zugreifen, wenn die geladene SWF-Datei die Methode `Security.allowDomain()` aufruft, um der Domäne der aufrufenden SWF-Datei Zugriffsberechtigungen zu erteilen. Sie können den Aufruf der `Security.allowDomain()`-Methode zur Hauptklassen-Konstruktormethode der geladenen SWF-Datei hinzufügen und dann die ladende SWF-Datei einen Event-Listener als Reaktion auf das `init`-Ereignis hinzufügen lassen, das von der Eigenschaft `contentLoaderInfo` des Loader-Objekts ausgelöst wird. Wenn dieses Ereignis ausgelöst wird, hat die geladene SWF-Datei die Methode `Security.allowDomain()` in der Konstruktormethode aufgerufen und Klassen in der geladenen SWF-Datei stehen der ladenden SWF-Datei zur Verfügung. Die ladende SWF-Datei kann Klassen aus der geladenen SWF-Datei abrufen, indem `Loader.contentLoaderInfo.applicationDomain.getDefinition()` oder Loader.contentLoaderInfo.applicationDomain.getQualifiedDefinitionNames() (Flash Player 11.3 und höher; AIR 3.3 und höher) aufgerufen wird.

Arbeiten mit Legacy-Inhalten

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In Flash Player 6 basiert die für bestimmte Flash Player-Einstellungen verwendete Domäne auf dem der Domäne der SWF-Datei nachgestellten Teil. Zu diesen Einstellungen zählen die Zugriffsberechtigungen auf Kamera und Mikrofon, Speicherkontingente und der Speicherplatz von permanenten gemeinsamen Objekten.

Wenn die Domäne einer SWF-Datei mehr als zwei Segmente enthält, beispielsweise www.example.com, wird das erste Segment der Domäne (www) entfernt und der restliche Teil der Domäne verwendet. In Flash Player 6 wird daher bei www.example.com und bei store.example.com die Domäne example.com als Domäne für diese Einstellungen verwendet. Genauso wird bei www.example.co.uk und store.example.co.uk die Domäne example.co.uk als Domäne für diese Einstellungen verwendet. Dies kann zu Problemen führen, durch die SWF-Dateien aus nicht verwandten Domänen, z. B. example1.co.uk und example2.co.uk, Zugriff auf die gleichen gemeinsamen Objekte haben.

In Flash Player 7 und späteren Versionen werden Player-Einstellungen in der Standardeinstellung entsprechend der exakten Domäne einer SWF-Datei ausgewählt. Beispiel: Eine SWF-Datei von www.example.com verwendet die Player-Einstellungen für www.example.com und eine SWF-Datei von store.example.com verwendet die Player-Einstellungen für store.example.com.

Bei einer SWF-Datei, die mit ActionScript 3.0 geschrieben wurde, verwendet Flash Player die exakten Domänen für die Player-Einstellungen, wenn `Security.exactSettings` auf `true` eingestellt ist (Standardeinstellung). Wenn dieser Parameter auf `false` festgelegt ist, verwendet Flash Player die Domäneneinstellungen aus Flash Player 6. Wenn Sie den Standardwert für `exactSettings` ändern, muss dies vor allen Ereignissen erfolgen, die von Flash Player das Auswählen der Player-Einstellungen anfordern, beispielsweise dem Verwenden einer Kamera oder eines Mikrofons oder dem Abrufen eines permanenten gemeinsamen Objekts.

Wenn Sie eine SWF-Datei der Version 6 veröffentlicht und daraus permanente gemeinsame Objekte erstellt haben, müssen Sie zum Abrufen dieser permanenten gemeinsamen Objekte aus einer SWF-Datei, die ActionScript 3.0 verwendet, `Security.exactSettings` auf `false` einstellen, bevor Sie `SharedObject.getLocal()` aufrufen.

Einstellen der LocalConnection-Berechtigungen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Mit der LocalConnection-Klasse können Sie Nachrichten von einer Flash Player- oder AIR-Anwendung zu einer anderen senden. LocalConnection-Objekte können nur mit Flash Player- oder AIR-Inhalt kommunizieren, der auf demselben Client ausgeführt wird, sie können aber in unterschiedlichen Anwendungen ausgeführt werden. Beispielsweise können eine in einem Browser ausgeführte SWF-Datei, eine in einem Projektor ausgeführte SWF-Datei und eine AIR-Anwendung alle über die LocalConnection-Klasse kommunizieren.

Jede LocalConnection-Kommunikation besteht aus einem Sender und einem Listener. Standardmäßig gestattet Flash Player die LocalConnection-Kommunikation zwischen Code, der in derselben Domäne ausgeführt wird. Bei Code, der in unterschiedlichen Sandboxen ausgeführt wird, muss der Listener dem Sender über die `LocalConnection.allowDomain()`-Methode eine Berechtigung erteilen. Die Zeichenfolge, die Sie als Argument an die Methode `LocalConnection.allowDomain()` übergeben, kann exakte Domänennamen, IP-Adressen sowie das Platzhalterzeichen `*` enthalten.

Die Form der Methode `allowDomain()` hat sich geändert. In den früheren Versionen (ActionScript 1.0 und 2.0) war `allowDomain` eine zu implementierende Rückrufmethode. In ActionScript 3.0 ist `allowDomain()` eine in die LocalConnection-Klasse eingebettete Methode. Aufgrund dieser Änderung ist die Methode `allowDomain()` jetzt mit `Security.allowDomain()` vergleichbar.

Mit der Eigenschaft `domain` der LocalConnection-Klasse kann eine SWF-Datei ihre Domäne ermitteln.

Steuern des externen URL-Zugriffs

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Für die externe Skripterstellung und den externen URL-Zugriff (durch die Verwendung von HTTP-URLs wie „mailto:“ usw.) werden die folgenden APIs verwendet:

- Die Funktion `flash.system.fscommand()`
- Die Methode `ExternalInterface.call()`
- Die Funktion `flash.net.navigateToURL()`

Bei Inhalt, der aus dem lokalen Dateisystem geladen wird, sind diese Methodenaufrufe nur erfolgreich, wenn der Code und die entsprechende Webseite (sofern vorhanden) sich in der Sandbox „local-trusted“ oder in der Sicherheits-Sandbox der AIR-Anwendung befinden. Aufrufe dieser Methoden schlagen fehl, wenn sich der Inhalt in der „local-with-networking“- oder der „local-with-filesystem“-Sandbox befindet.

Bei nicht lokal geladenem Inhalt können alle diese APIs abhängig vom Wert des AllowScriptAccess-Parameters (siehe unten) mit der Webseite kommunizieren, in der sie eingebettet sind. Die Funktion `flash.net.navigateToURL()` kann zudem mit allen geöffneten Browserfenstern oder Frames kommunizieren. Sie ist nicht auf die Seite beschränkt, in der die SWF-Datei eingebettet ist. Weitere Informationen zu dieser Funktion finden Sie unter „Verwenden der navigateToURL()-Funktion“ auf Seite 1151.

Der `AllowScriptAccess`-Parameter in dem HTML-Code, der eine SWF-Datei lädt, steuert den externen URL-Zugriff aus einer SWF-Datei heraus. Legen Sie diesen Parameter in den Tags PARAM oder EMBED fest. Wenn Sie für `AllowScriptAccess` keinen Wert angeben, können die SWF-Datei und die HTML-Seite nur miteinander kommunizieren, wenn sie aus der gleichen Domäne stammen.

Der Parameter `AllowScriptAccess` kann folgende Werte annehmen: `„always"`, `„sameDomain"` oder `„never"`.

- Wenn `AllowScriptAccess` auf `„always"` festgelegt ist, kann die SWF-Datei mit der HTML-Seite, in die sie eingebettet ist, auch dann kommunizieren, wenn die SWF-Datei aus einer anderen Domäne als die HTML-Seite stammt.
- Wenn `AllowScriptAccess` auf `„sameDomain"` festgelegt ist, kann die SWF-Datei mit der HTML-Seite, in die sie eingebettet ist, nur dann kommunizieren, wenn die SWF-Datei aus der gleichen Domäne wie die HTML-Seite stammt. Dies ist der Standardwert für `AllowScriptAccess`. Verwenden Sie diese Einstellung, oder geben Sie keinen Wert für `AllowScriptAccess` an, um zu verhindern, dass eine SWF-Datei in einer Domäne auf eine HTML-Seite aus einer anderen Domäne zugreift.
- Wenn `AllowScriptAccess` auf `„never"` festgelegt ist, kann die SWF-Datei mit keiner HTML-Seite kommunizieren. Die Verwendung dieses Werts ist seit der Veröffentlichung von Adobe Flash CS4 Professional veraltet. Von der Verwendung dieses Werts wird abgeraten und er sollte nicht erforderlich sein, wenn Sie keine nicht vertrauenswürdigen SWF-Dateien von Ihrer Domäne aus bereitstellen. Falls Sie nicht vertrauenswürdige SWF-Dateien bereitstellen müssen, empfiehlt es sich, eine separate Unterdomäne zu erstellen und alle nicht vertrauenswürdigen Inhalte dort abzulegen.

Das folgende Beispiel veranschaulicht, wie Sie den Tag `AllowScriptAccess` in einer HTML-Seite so einstellen, dass ein externer URL-Zugriff auf eine andere Domäne möglich ist:

```
<object id='MyMovie.swf' classid='clsid:D27CDB6E-AE6D-11cf-96B8-444553540000'
codebase='http://download.adobe.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,0,0'
height='100%' width='100%'>
<param name='AllowScriptAccess' value='always'/>
<param name='src' value=''MyMovie.swf'/>
<embed name='MyMovie.swf' pluginspage='http://www.adobe.com/go/getflashplayer'
src='MyMovie.swf' height='100%' width='100%' AllowScriptAccess='never'/>
</object>
```

Verwenden der navigateToURL()-Funktion

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Zusätzlich zu dem Parameter `allowScriptAccess` zum Angeben von Sicherheitseinstellungen (siehe oben) verfügt die Funktion `navigateToURL()` über einen optionalen zweiten Parameter: `target`. Mit dem Parameter `target` können Sie ein HTML-Fenster oder einen Frame angeben, an das bzw. den die URL-Anforderung gesendet werden soll. Für diese Anforderungen gelten zusätzliche Sicherheitseinschränkungen. Diese variieren in Abhängigkeit davon, ob `navigateToURL()` als Scripting- oder Non-Scripting-Anweisung verwendet wird.

Für Scripting-Anweisungen (z. B. `navigateToURL("javascript: alert('Hello from Flash Player.')")`) gelten die folgenden Regeln.

- Wenn es sich bei der SWF-Datei um eine lokal vertrauenswürdige Datei handelt, ist die Anforderung erfolgreich.
- Wenn als Ziel die HTML-Seite verwendet wird, in die die SWF-Datei eingebettet ist, gelten die oben erläuterten `allowScriptAccess`-Regeln.
- Wenn das Ziel Inhalte enthält, die aus der gleichen Domäne wie die SWF-Datei geladen wurden, ist die Anforderung erfolgreich.
- Wenn das Ziel Inhalte enthält, die aus einer anderen Domäne als die SWF-Datei geladen wurden, und die beiden vorherigen Bedingungen nicht erfüllt sind, schlägt die Anforderung fehl.

Bei Non-Scripting-Anweisungen (z. B. HTTP, HTTPS und `mailto:`) schlägt die Anforderung fehl, wenn die folgenden Bedingungen zutreffen:

- Das Ziel entspricht einem der speziellen Schlüsselwörter `"_top"` oder `"_parent"`; UND
- Die SWF-Datei ist Bestandteil einer Webseite in einer anderen Domäne; UND
- Die SWF-Datei wurde mit einem Wert für `allowScriptAccess` eingebettet, der nicht „`always"` lautet.

Weitere Informationen

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Weitere Informationen zum externen URL-Zugriff finden Sie in den folgenden Abschnitten im *ActionScript 3.0-Referenzhandbuch für die Adobe Flash-Plattform*:

- Die flash.system.fscommand()-Funktion
- Die call()-Methode der ExternalInterface-Klasse
- Die `flash.net.navigateToURL()`-Funktion

Gemeinsame Objekte

Flash Player 9 und höher, Adobe AIR 1.0 und höher

In Flash Player können Sie *gemeinsame Objekte* verwenden, bei denen es sich um ActionScript-Objekte handelt, die sich außerhalb einer SWF-Datei befinden, entweder lokal auf dem Dateisystem eines Benutzers oder remote auf einem RTMP-Server. Gemeinsame Objekte sind, wie andere Medien in Flash Player, in Sicherheits-Sandboxen partitioniert. Das Sandbox-Modell für gemeinsame Objekte ist jedoch etwas anders, da gemeinsame Objekte keine Ressourcen darstellen, auf die domänenübergreifend zugegriffen wird. Stattdessen werden gemeinsame Objekte immer aus einem

gemeinsamen Objektspeicher abgerufen, der nur für die Domäne einer SWF-Datei gilt, die Methoden der SharedObject-Klasse abruft. Ein Speicher für gemeinsame Objekte ist im Allgemeinen noch spezifischer als die Domäne einer SWF-Datei. Standardmäßig verwendet jede SWF-Datei einen Speicher für gemeinsame Objekte, der für die gesamte Ursprungs-URL spezifisch ist. Weitere Informationen zu gemeinsamen Objekten finden Sie unter „Gemeinsame Objekte“ auf Seite 747.

Eine SWF-Datei kann den Parameter `localPath` der Methoden `SharedObject.getLocal()` und `SharedObject.getRemote()` verwenden, um einem Speicher für gemeinsame Objekte nur einen Teil der URL zuzuweisen. Auf diese Weise kann die SWF-Datei das gemeinsame Nutzen mit anderen SWF-Dateien von anderen URLs zulassen. Auch wenn Sie `'/'` als `localPath`-Parameter übergeben, wird immer noch ein Speicherbereich für gemeinsame Objekte angegeben, der nur in der eigenen Domäne gültig ist.

Benutzer können den Zugriff auf gemeinsam genutzte Objekte über das Flash Player-Dialogfeld „Einstellungen“ oder über den Einstellungsmanager einschränken. Für gemeinsam genutzte Objekte gilt standardmäßig eine Obergrenze von 100 KB an Daten pro Domäne. Administratoren und Benutzer können auch Einschränkungen für Schreibberechtigungen auf das Dateisystem festlegen. Weitere Informationen finden Sie unter „Kontrolloptionen für Administratoren“ auf Seite 1122 und „Kontrolloptionen für Benutzer“ auf Seite 1124.

Durch Einstellen von `true` für den Parameter `secure` der `SharedObject.getLocal()`- oder `SharedObject.getRemote()`-Methode können Sie angeben, dass ein gemeinsames Objekt sicher ist. Folgendes müssen Sie beim Parameter `secure` beachten:

- Falls dieser Parameter auf `true` gesetzt wird, erstellt Flash Player ein neues sicheres gemeinsam genutztes Objekt oder ruft einen Verweis auf ein vorhandenes sicheres gemeinsam genutztes Objekt ab. Dieses sichere gemeinsam genutzte Objekt kann nur durch SWF-Dateien gelesen oder geschrieben werden, die über HTTPS bereitgestellt werden und `SharedObject.getLocal()` mit der Einstellung `true` für den `secure`-Parameter aufrufen.
- Wenn dieser Parameter auf `false` gesetzt wird, erstellt Flash Player ein neues gemeinsames Objekt oder ruft den Verweis auf ein vorhandenes gemeinsames Objekt ab, das durch SWF-Dateien gelesen oder geschrieben werden kann, die über andere Verbindungen als HTTPS bereitgestellt werden.

Wenn die aufrufende SWF-Datei nicht von einer HTTPS-URL stammt, löst das Einstellen von `true` für den Parameter `secure` der `SharedObject.getLocal()`- oder `SharedObject.getRemote()`-Methode eine SecurityError-Ausnahme aus.

Die Auswahl eines Speichers für ein gemeinsames Objekt basiert auf der Ursprungs-URL der SWF-Datei. Dies gilt auch in den zwei Situationen, in denen eine SWF-Datei nicht von einer einfachen URL stammt. beim importierten Laden und beim dynamischen Laden. Importiertes Laden bezieht sich auf eine Situation, in der Sie eine SWF-Datei laden und in der die `LoaderContext.securityDomain`-Eigenschaft auf `SecurityDomain.currentDomain` eingestellt ist. In einer solchen Situation hat die geladene SWF-Datei eine Pseudo-URL, die mit der Domäne der ladenden SWF-Datei beginnt und dann die tatsächliche Ursprungs-URL angibt. Dynamisches Laden bezieht sich auf das Laden einer SWF-Datei mit der `Loader.loadBytes()`-Methode. In dieser Situation hat die geladene SWF-Datei eine Pseudo-URL, die mit der vollständigen URL der ladenden SWF-Datei beginnt, gefolgt von einer Integer-ID. In beiden Fällen, sowohl beim importiertem Laden als auch beim dynamischen Laden, kann die Pseudo-URL der SWF-Datei mit der Eigenschaft `LoaderInfo.url` ermittelt werden. Die Pseudo-URL wird bei der Auswahl eines Speichers für gemeinsame Objekte genau so wie eine reale URL behandelt. Sie können einen Parameter `localPath` für ein gemeinsames Objekt angeben, der die Pseudo-URL ganz oder teilweise verwendet.

Benutzer und Administratoren können die Verwendung von *gemeinsamen Objekte von Dritten* deaktivieren. Dies ist die Verwendung von gemeinsamen Objekten durch eine SWF-Datei, die in einem Webbrowser ausgeführt wird, wenn die Ursprungs-URL dieser SWF-Datei von einer anderen Domäne stammt als die URL, die in der Adressleiste des Browsers angezeigt wird. Benutzer und Administratoren können die Verwendung von gemeinsamen Objekten von Dritten aus Sicherheitsgründen deaktivieren, um auf diese Weise eine domänenübergreifende Verfolgung zu verhindern. Um diese Einschränkung zu vermeiden, können Sie sicherstellen, dass eine SWF-Datei, die gemeinsame

Objekte verwendet, nur innerhalb der HTML-Seitenstrukturen geladen wird. Dies stellt sicher, dass die SWF-Datei aus der gleichen Domäne stammt, die in der Adressleiste des Browsers angezeigt wird. Wenn Sie versuchen, gemeinsame Objekte aus einer SWF-Datei von Dritten zu verwenden, und das Verwenden von gemeinsamen Objekten von Dritten deaktiviert ist, geben die `SharedObject.getLocal()`- und `SharedObject.getRemote()`-Methoden `null` zurück. Weitere Informationen finden Sie unter www.adobe.com/de/products/flashplayer/articles/thirdpartylso.

Zugriff auf Kamera, Mikrofon, Zwischenablage, Maus und Tastatur

Flash Player 9 und höher, Adobe AIR 1.0 und höher

Wenn eine SWF-Datei versucht, mit den Methoden `Camera.get()` oder `Microphone.get()` auf die Kamera oder das Mikrofon eines Benutzers zuzugreifen, öffnet Flash Player ein Datenschutzdialogfeld, in dem der Benutzer den Zugriff auf die Kamera oder das Mikrofon gestatten oder verweigern kann. Benutzer und Administratoren können den Kamerazugriff auch auf Site- oder globaler Ebene deaktivieren. Hierzu werden Steuerelemente in der Datei „mms.cfg“, der Einstellungs-UI und dem Einstellungsmanager verwendet (siehe „Kontrolloptionen für Administratoren“ auf Seite 1122 und „Kontrolloptionen für Benutzer“ auf Seite 1124). Mit Benutzereinschränkungen geben die Methoden `Camera.get()` und `Microphone.get()` den Wert `null` zurück. Mit der Eigenschaft `Capabilities.avHardwareDisable` können Sie festlegen, ob der Zugriff auf Kamera und Mikrofon von administrativer Seite verboten (`true`) oder gestattet ist (`false`).

Die Methode `System.setClipboard()` ermöglicht einer SWF-Datei, den Inhalt der Zwischenablage durch einen unformatierten Textstring zu ersetzen. Dies stellt kein Sicherheitsrisiko dar. Zum Schutz vor dem Risiko, dass Kennwörter und andere wichtige Daten ausgeschnitten oder aus der Zwischenablage kopiert werden, gibt es keine entsprechende `getClipboard()`-Methode.

Eine in Flash Player ausgeführte Anwendung kann nur die innerhalb ihres Fokus auftretenden Tastatur- und Mausereignisse überwachen. In Flash Player ausgeführter Inhalt kann Tastatur- und Mausereignisse in anderen Anwendungen nicht erkennen.

AIR-Sicherheit

Adobe AIR 1.0 und höher

Sicherheitsgrundlagen für AIR

Adobe AIR 1.0 und höher

AIR-Anwendungen werden mit denselben Sicherheitseinschränkungen wie native Anwendungen ausgeführt. Im Allgemeinen haben AIR-Anwendungen genau wie native Anwendungen umfassenden Zugriff auf Betriebssystemfunktionen wie Lesen von Dateien, Schreiben in Dateien, Starten von Anwendungen, Zeichnen auf dem Bildschirm und Kommunikation mit dem Netzwerk. Betriebssystemeinschränkungen, die für native Anwendungen gelten, zum Beispiel benutzerspezifische Berechtigungen, gelten gleichermaßen für AIR-Anwendungen.

Das Adobe® AIR®-Sicherheitsmodell ist zwar eine Weiterentwicklung des Adobe® Flash® Player-Sicherheitsmodells, der Sicherheitsvertrag unterscheidet sich jedoch von dem Sicherheitsvertrag, der für Inhalte in einem Browser gilt. Dieser Vertrag bietet Entwicklern ein sicheres Mittel umfangreicherer Funktionen mit Freiheiten, die für browserbasierte Anwendungen nicht geeignet sind.

AIR-Anwendungen werden entweder mit kompiliertem Bytecode (SWF-Inhalt) oder mit interpretiertem Skript (JavaScript, HTML) geschrieben, sodass die Laufzeitumgebung die Speicherverwaltung übernimmt. Auf diese Weise wird das Risiko verringert, dass AIR-Anwendungen durch Pufferüberlauf, Speicherbeschädigung und andere Sicherheitslücken, die mit der Speicherverwaltung zu tun haben, beeinträchtigt werden. Diese gehören zu den häufigsten Sicherheitsrisiken, die in nativem Code geschriebene Desktopanwendungen betreffen.

Installation und Updates

Adobe AIR 1.0 und höher

AIR-Anwendungen werden über AIR-Installationsdateien mit der `air`-Erweiterung oder über native Installationsprogramme verteilt. Letztere verwenden das Dateiformat und die Dateierweiterung der nativen Plattform. Das Format eines nativen Windows-Installationsprogramms ist beispielsweise eine EXE-Datei, während das native Format für Android eine APK-Datei ist.

Wenn Adobe AIR installiert ist und eine AIR-Installationsprogrammdatei geöffnet wird, verwaltet die AIR-Laufzeitumgebung den Installationsprozess. Bei Verwendung eines nativen Installationsprogramms verwaltet das Betriebssystem den Installationsprozess.

Hinweis: *Entwickler können bei Verwendung des AIR-Dateiformats eine Version, einen Anwendungsnamen und eine Veröffentlicherquelle angeben, der anfängliche Ablauf der Anwendungsinstallation kann jedoch nicht geändert werden. Diese Einschränkung hat für die Benutzer Vorteile, da alle AIR-Anwendungen einen sicheren, effizienten und konsistenten Installationsablauf verwenden, der von der Laufzeitumgebung verwaltet wird. Wenn die Anwendung modifiziert werden muss, kann dies bei der ersten Ausführung der Anwendung erfolgen.*

Installationsort der Laufzeitumgebung

Adobe AIR 1.0 und höher

Für AIR-Anwendungen, die das AIR-Dateiformat verwenden, muss zunächst die Laufzeitumgebung auf dem Computer des Benutzers installiert werden, so wie für SWF-Dateien das Browser-Plug-In Flash Player installiert sein muss.

Die Laufzeitumgebung wird auf Desktopcomputern am folgenden Speicherort installiert:

- Mac OS: `/Library/Frameworks/`
- Windows: `C:\Programme\Gemeinsame Dateien\Adobe AIR`
- Linux: `/opt/Adobe AIR/`

Wenn unter Mac OS eine aktualisierte Version der Anwendung installiert werden soll, benötigt der Benutzer die entsprechenden Systemberechtigungen, um im Anwendungsverzeichnis zu installieren. Unter Windows und Linux muss der Benutzer über Administratorberechtigungen verfügen.

Hinweis: *Unter iOS wird die AIR-Laufzeitumgebung nicht separat installiert; bei jeder AIR-Anwendung handelt es sich um eine eigenständige Anwendung.*

Die Laufzeitumgebung kann auf zwei Arten installiert werden: mit der nahtlosen Installationsfunktion (direkte Installation von einem Webbrowser) oder manuell. AIR-Anwendungen, die als native Installationsprogramme verpackt sind, können die AIR-Laufzeitumgebung auch als Teil ihres normalen Installationsvorgangs installieren. (Für diese Verbreitung der AIR-Laufzeitumgebung ist eine Redistributionsvereinbarung mit Adobe erforderlich.)

Nahtlose Installation (Laufzeitumgebung und Anwendung)

Adobe AIR 1.0 und höher

Die nahtlose Installationsfunktion stellt Entwicklern eine Installationsmöglichkeit für Benutzer, die Adobe AIR noch nicht installiert haben, zur Verfügung. Bei der nahtlosen Installationsmethode erstellt der Entwickler eine SWF-Datei, die die Anwendung zur Installation anzeigt. Wenn ein Benutzer auf die SWF-Datei klickt, um die Anwendung zu installieren, sucht die SWF-Datei nach der Laufzeitumgebung. Wenn die Laufzeitumgebung nicht gefunden wird, wird sie installiert und die Laufzeitumgebung wird sofort mit dem Installationsvorgang für die Anwendung des Entwicklers aktiviert.

Manuelle Installation

Adobe AIR 1.0 und höher

Alternativ dazu kann der Benutzer die Laufzeitumgebung vor dem Öffnen einer AIR-Datei manuell herunterladen und installieren. Der Entwickler kann eine AIR-Datei dann auf verschiedene Weisen verteilen (zum Beispiel per E-Mail oder über einen HTML-Link auf einer Website). Wenn die AIR-Datei geöffnet wird, beginnt die Laufzeitumgebung mit der Verarbeitung der Anwendungsinstallation.

Ablauf der Anwendungsinstallation

Adobe AIR 1.0 und höher

Beim AIR-Sicherheitsmodell können Benutzer entscheiden, ob sie eine AIR-Anwendung installieren möchten. Die AIR-Installation bietet verschiedene Verbesserungen gegenüber Installationsmethoden für native Anwendungen, die den Benutzern diese Entscheidung leichter macht:

- Die Laufzeitumgebung ermöglicht den konsistenten Installationsablauf auf allen Betriebssystemen, auch wenn die AIR-Anwendung über einen Link in einem Webbrowser installiert wird. In den Installationsabläufen der meisten nativen Anwendungen ist die Bereitstellung von Sicherheitsinformationen dagegen abhängig vom Browser oder anderen Anwendungen, falls überhaupt Sicherheitsinformationen verfügbar sind.
- Bei der AIR-Anwendungsinstallation wird die Quelle der Anwendung identifiziert und es wird festgestellt, welche Berechtigungen für die Anwendung verfügbar sind (sofern der Benutzer die Fortsetzung der Installation zulässt).
- Die Laufzeitumgebung verwaltet den Installationsprozess einer AIR-Anwendung. Der von der Laufzeitumgebung verwendete Installationsprozess kann von einer AIR-Anwendung nicht geändert werden.

Im Allgemeinen sollten Benutzer keine Desktopanwendungen installieren, die von einer nicht vertrauenswürdigen oder nicht überprüfbaren Quelle stammen. Die Sicherheitsprüfung für native Anwendungen gilt ebenso für AIR-Anwendungen wie für andere installierbare Anwendungen.

Anwendungsziel

Adobe AIR 1.0 und höher

Das Installationsverzeichnis kann mit einer der beiden folgenden Optionen festgelegt werden:

1. Der Benutzer kann das Ziel während der Installation anpassen. Die Anwendung wird am vom Benutzer angegebenen Speicherort installiert.
2. Wenn der Benutzer das Installationsziel nicht ändert, wird die Anwendung unter dem von der Laufzeitumgebung festgelegten Standardpfad installiert:
   - Mac OS: `~/Applications/`

- Windows XP und älter: `C:\Programme\`
- Windows Vista: `~/Apps/`
- Linux: /opt/

Wenn der Entwickler in der Anwendungsdeskriptordatei eine `installFolder`-Einstellung angibt, wird die Anwendung in einem Unterpfad dieses Verzeichnisses installiert.

Das AIR-Dateisystem

Adobe AIR 1.0 und höher

Beim Installationsprozess für AIR-Anwendungen werden alle Dateien, die der Entwickler in die AIR-Installationsprogrammdatei einbezogen hat, auf den lokalen Computer des Benutzers kopiert. Die installierte Anwendung besteht aus Folgendem:

- Windows: ein Verzeichnis, das alle Dateien enthält, die in die AIR-Installationsprogrammdatei einbezogen wurden. Während der Installation der AIR-Anwendung erstellt die Laufzeitumgebung auch eine exe-Datei.
- Linux: Ein Verzeichnis, das alle Dateien enthält, die in die AIR-Installationsprogrammdatei einbezogen wurden. Während der Installation der AIR-Anwendung erstellt die Laufzeitumgebung auch eine bin-Datei.
- Mac OS: eine `app`-Datei, die den gesamten Inhalt der AIR-Installationsprogrammdatei enthält. Sie kann mit der Option „Show Package Contents“ (Paketinhalt anzeigen) im Finder überprüft werden. Die Laufzeitumgebung erstellt diese app-Datei als Teil der Installation einer AIR-Anwendung.

Eine AIR-Anwendung wird folgendermaßen ausgeführt:

- Windows: Durch Ausführen der .exe-Datei im Installationsordner oder über eine Verknüpfung, die dieser Datei entspricht (zum Beispiel eine Verknüpfung im Start-Menü oder auf dem Desktop).
- Linux: Durch Starten der .bin-Datei im Installationsordner, durch Auswählen der Anwendung aus dem Anwendungsmenü oder durch Ausführen eines Alias oder einer Desktopverknüpfung.
- Mac OS: Durch Ausführen der .app-Datei oder eines Alias, der auf die Datei verweist.

Das Dateisystem der Anwendung enthält auch Unterverzeichnisse, die mit der Funktion der Anwendung zu tun haben. Zum Beispiel werden Informationen, die in verschlüsselten lokalen Speicher geschrieben werden, in einem Unterverzeichnis unter einem Verzeichnis, das nach dem Anwendungsbezeichner der Anwendung benannt ist, gespeichert.

AIR-Anwendungsspeicher

Adobe AIR 1.0 und höher

AIR-Anwendungen haben die Berechtigung, auf jeden Speicherort auf der Festplatte des Benutzers zu schreiben. Entwicklern wird jedoch empfohlen, den Pfad `app-storage:/` als lokalen Speicher für ihre Anwendung zu nutzen. Dateien, die von einer Anwendung in `app-storage:/` geschrieben werden, befinden sich an einem Standardspeicherort, der sich nach dem Betriebssystem des Benutzers richtet:

- Unter Mac OS: der Speicherordner einer Anwendung variiert je nach AIR-Version:
  - **AIR 3.2 und früher** - `<appDaten>/<appId>/Local Store/`, wobei `<appDaten>` der Voreinstellungen-Ordner des Benutzers ist, normalerweise: `/Benutzer/<Benutzername>/Library/Preferences`
  - **AIR 3.3 und höher** - `<Pfad>/Library/Application Support/<appID>/Local Store`, wobei `<Pfad>` entweder `/Benutzer/<Benutzername>/Library/Containers/<bundle-id>/Data` (Sandbox-Umgebung) oder `/Benutzer/<Benutzername>` ist (bei Ausführung außerhalb einer Sandbox-Umgebung)

- Windows: Das Speicherverzeichnis einer Anwendung ist `<appData>\<appId>\Local Store\`, wobei `<appData>` der spezielle CSIDL_APPDATA-Ordner des Benutzers ist, normalerweise `C:\Dokumente und Einstellungen\<Benutzer>\Anwendungsdaten`
- Linux: `<appData>/<appID>/Local Store/`wobei `<appData>/home/<Benutzer>/.appdata` ist

Sie können über die `air.File.applicationStorageDirectory`-Eigenschaft auf das Anwendungsspeicherverzeichnis zugreifen. Auf den Verzeichnisinhalt können Sie mit der `resolvePath()`-Methode der File-Klasse zugreifen. Ausführliche Informationen finden Sie unter „Arbeiten mit dem Dateisystem“ auf Seite 695.

Aktualisieren von Adobe AIR

Adobe AIR 1.0 und höher

Wenn der Benutzer eine AIR-Anwendung installiert, die eine aktualisierte Version der Laufzeitumgebung benötigt, installiert die Laufzeitumgebung das erforderliche Update automatisch.

Für die Aktualisierung der Laufzeitumgebung muss ein Benutzer über Administratorberechtigungen für den Computer verfügen.

Aktualisieren von AIR-Anwendungen

Adobe AIR 1.0 und höher

Die Entwicklung und Bereitstellung von Softwareupdates stellt bezüglich der Sicherheit bei nativen Code-Anwendungen eine der größten Herausforderungen dar. Die AIR-API verbessert dies: die `Updater.update()`-Methode kann beim Starten aufgerufen werden, um einen Remote-Speicherort auf eine AIR-Datei zu überprüfen. Wenn ein Update ausgeführt werden sollte, wird die AIR-Datei heruntergeladen und installiert und die Anwendung wird neu gestartet. Entwickler können mit dieser Klasse nicht nur neue Funktionen bereitstellen, sondern auch auf potenzielle Sicherheitslücken reagieren.

Mit der Updater-Klasse können nur Anwendungen aktualisiert werden, die als AIR-Dateien verteilt werden. Anwendungen, die in nativer Form verteilt werden, müssen die Aktualisierungsfunktion des nativen Betriebssystems verwenden, sofern vorhanden.

Hinweis: *Entwickler können die Version einer Anwendung angeben, indem sie die versionNumber-Eigenschaft der Anwendungsdeskriptordatei festlegen.*

Deinstallieren von AIR-Anwendungen

Adobe AIR 1.0 und höher

Beim Entfernen einer AIR-Anwendung werden alle Dateien im Anwendungsverzeichnis entfernt. Es werden jedoch nicht alle Dateien entfernt, die die Anwendung möglicherweise außerhalb des Anwendungsverzeichnisses geschrieben hat. Änderungen, die die AIR-Anwendung an Dateien außerhalb des Anwendungsverzeichnisses vorgenommen hat, werden beim Entfernen der AIR-Anwendung nicht zurückgenommen.

Einstellungen in der Windows-Registrierung für Administratoren

Adobe AIR 1.0 und höher

Unter Windows können Administratoren den Computer so konfigurieren, dass AIR-Anwendungsinstallationen und Updates der Laufzeitumgebung verhindert (oder zugelassen) werden. Diese Einstellungen sind unter dem folgenden Schlüssel in der Windows-Registrierung enthalten: HKLM\Software\Policies\Adobe\AIR. Dazu gehört Folgendes:

Einstellung in der Registrierung	Beschreibung
AppInstallDisabled	Legt fest, dass die Installation und Deinstallation der AIR-Anwendung zulässig sind. Für „zulässig“ auf 0, für „nicht zulässig“ auf 1 gesetzt.
UntrustedAppInstallDisabled	Legt fest, dass die Installation von nicht vertrauenswürdigen AIR-Anwendungen zulässig ist (Anwendungen, die kein vertrauenswürdiges Zertifikat besitzen). Für „zulässig“ auf 0, für „nicht zulässig“ auf 1 gesetzt.
UpdateDisabled	Legt fest, dass die Aktualisierung der Laufzeitumgebung zulässig ist, entweder als Hintergrund-Task oder als Teil einer expliziten Installation. Für „zulässig“ auf 0, für „nicht zulässig“ auf 1 gesetzt.

HTML-Sicherheit in Adobe AIR

Adobe AIR 1.0 und höher

In diesem Thema wird die HTML-Sicherheitsarchitektur von AIR beschrieben. Sie erfahren, wie Sie Inlineframes (iframes), Frames und die Sandboxbrücke verwenden, um HTML-Anwendungen einzurichten und HTML-Inhalt sicher in SWF-Anwendungen zu integrieren.

Die Laufzeitumgebung erzwingt Regeln und stellt Mechanismen für die Vermeidung möglicher Sicherheitslücken in HTML und JavaScript bereit. Es gelten dieselben Regeln, unabhängig davon, ob Ihre Anwendung hauptsächlich in JavaScript geschrieben ist oder ob Sie die HTML- und JavaScript-Inhalte in eine SWF-basierte Anwendung laden. Inhalte in der Anwendungs-Sandbox und in anwendungsfremden Sicherheits-Sandboxen verfügen über unterschiedliche Berechtigungen. Beim Laden von Inhalten in einen Inlineframe oder Frame stellt die Laufzeitumgebung einen sicheren *Sandboxbrücken*-Mechanismus bereit, der es dem Inhalt im Inlineframe oder Frame ermöglicht, sicher mit Inhalt in der Anwendungs-Sandbox zu kommunizieren.

Das AIR-SDK bietet drei Klassen zum Rendern von HTML-Inhalt.

Die HTMLLoader-Klasse ermöglicht eine enge Integration zwischen JavaScript-Code und den AIR-APIs.

Die StageWebView-Klasse ist eine HTML-Render-Klasse, die nur sehr eingeschränkte Integration mit der AIR-Hostanwendung bietet. Von der StageWebView-Klasse geladener Inhalt wird nie in der Sicherheits-Sandbox der Anwendung platziert und kann nicht auf Daten zugreifen oder Funktionen aufrufen, die sich in der AIR-Hostanwendung befinden. Auf Desktop-Plattformen verwendet die StageWebView-Klasse die integrierte HTML-Engine von AIR, die auf dem Webkit basiert, das auch von der HTMLLoader-Klasse verwendet wird. Auf mobilen Plattformen verwendet die StageWebView-Klasse die HTML-Steuerung des Betriebssystems. Deshalb weist die StageWebView-Klasse auf mobilen Plattformen dieselben Sicherheitsaspekte und Schwachstellen auf wie der Webbrowser des Systems.

Die TextField-Klasse kann HTML-Textstrings anzeigen. JavaScript kann nicht ausgeführt werden, aber der Text kann Links und aus externen Quellen geladene Bilder enthalten.

Weitere Informationen finden Sie unter „Vermeiden von sicherheitsbezogenen JavaScript-Fehlern“ auf Seite 1049.

Überblick über die Konfiguration HTML-basierter Anwendungen

Adobe AIR 1.0 und höher

Frames und Inlineframes bieten eine praktische Struktur für HTML-Inhalte in AIR. Frames sind eine Möglichkeit, sowohl die Datenkonsistenz zu wahren als auch sicher mit Remote-Inhalten zu arbeiten.

Da HTML in AIR die normale, seitenbasierte Organisation beibehält, wird die HTML-Umgebung vollständig aktualisiert, wenn der obere Frame des HTML-Inhalts zu einer anderen Seite „navigiert“. Sie können Frames und Inlineframes verwenden, um die permanente Datenspeicherung in AIR zu erhalten, so wie für eine Webanwendung, die in einem Browser ausgeführt wird. Definieren Sie Ihre Hauptanwendungsobjekte im obersten Frame, damit sie dauerhaft bestehen, solange Sie nicht zulassen, dass der Frame zu einer neuen Seite navigiert. Verwenden Sie untergeordnete Frames oder Inlineframes, um die Übergangsteile der Anwendung zu laden. (Es gibt verschiedene andere Möglichkeiten, Daten dauerhaft zu speichern, die zusätzlich zu oder anstelle von Frames verwendet werden können. Dazu gehören Cookies, lokale freigegebene Objekte, lokaler Dateispeicher, der verschlüsselte Dateispeicher und lokaler Datenbankspeicher.)

Die Grenze zwischen ausführbarem Code und Daten ist auch bei HTML in AIR verschwommen. Deshalb stellt AIR Inhalt im obersten Frame der HTML-Umgebung in die Anwendungs-Sandbox. Nach dem `load`-Seitenereignis beschränkt AIR alle Operationen, zum Beispiel `eval()`, die einen Textstring in eine ausführbares Objekt konvertieren können. Diese Beschränkung wird auch dann erzwungen, wenn eine Anwendung keine Remote-Inhalte lädt. Damit HTML-Inhalt diese eingeschränkten Operationen ausführen kann, müssen Sie Frames oder Inlineframes verwenden, um den Inhalt in einer anwendungsfremden Sandbox zu platzieren. (Die Ausführung von Inhalten in einem untergeordneten Frame in einer Sandbox kann erforderlich sein, wenn bestimmtes JavaScript-Anwendungs-Framework verwendet wird, das auf der `eval()`-Funktion basiert.) Eine vollständige Liste der Beschränkungen für JavaScript in der Anwendungs-Sandbox finden Sie unter „Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen“ auf Seite 1161.

Da HTML in AIR die Möglichkeit beibehält, remote, potenziell unsichere Inhalte zu laden, erzwingt AIR eine Richtlinie derselben Herkunft, die verhindert, dass Inhalt in einer Domäne mit Inhalt in einer anderen Domäne interagiert. Um Interaktionen zwischen Anwendungsinhalten und Inhalten in anderen Domänen zuzulassen, können Sie eine „Brücke“ als Schnittstelle zwischen einem übergeordneten und einem untergeordnetem Frame einrichten.

Einrichten einer hierarchischen Sandbox-Beziehung

Adobe AIR 1.0 und höher

AIR fügt den frame- und iframe-HTML-Elementen die Attribute `sandboxRoot` und `documentRoot` hinzu. Mithilfe dieser Attribute kann Anwendungsinhalt so behandelt werden, als käme er aus einer anderen Domäne:

Attribut	Beschreibung
sandboxRoot	Die URL, mit der die Sandbox und die Domäne, in der der Frame-Inhalt platziert wird, bestimmt wird. Es muss das URL-Schema `file:`, `http:` oder `https:` verwendet werden.
documentRoot	Die URL, von der der Frame-Inhalt geladen wird. Es muss das URL-Schema `file:`, `app:` oder `app-storage:` verwendet werden.

Im folgenden Beispiel wird Inhalt im Sandbox-Unterverzeichnis der Anwendung für die Ausführung in der Remote-Sandbox und der Domäne www.example.com domain bestimmt:

```
<iframe
    src="ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

Einrichten einer Brücke zwischen übergeordneten und untergeordneten Frames in unterschiedlichen Sandboxen oder Domänen

Adobe AIR 1.0 und höher

AIR fügt die `childSandboxBridge`-Eigenschaft und die `parentSandboxBridge`-Eigenschaft dem `window`-Objekt jedes untergeordneten Frames hinzu. Mit diesen Eigenschaften können Sie Brücken definieren, die als Schnittstellen zwischen einem übergeordneten und einem untergeordneten Frame dienen. Jede Brücke verläuft in eine Richtung:

`childSandboxBridge` – Die `childSandboxBridge`-Eigenschaft ermöglicht dem untergeordneten Frame, eine Schnittstelle zum Inhalt im übergeordneten Frame zu öffnen. Um eine Schnittstelle zu öffnen, stellen Sie die `childSandbox`-Eigenschaft für eine Funktion oder ein Objekt im untergeordneten Frame ein. Sie können dann vom Inhalt im übergeordneten Frame aus auf das Objekt oder die Funktion zugreifen. Im folgenden Beispiel wird veranschaulicht, wie ein Skript, das in einem untergeordneten Frame ausgeführt wird, ein Objekt, das eine Funktion und eine Eigenschaft enthält, für den übergeordneten Frame öffnen kann:

```
var interface = {};
interface.calculatePrice = function(){
    return .45 + 1.20;
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Wenn sich dieser untergeordnete Inhalt in einem Inlineframe mit der zugewiesenen `id"child"` befindet, können Sie vom übergeordneten Inhalt auf die Schnittstelle zugreifen, indem Sie die `childSandboxBridge`-Eigenschaft des Frames lesen:

```
var childInterface = document.getElementById("child").childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces "1.65"
air.trace(childInterface.storeID)); //traces "abc"
```

`parentSandboxBridge` – Die `parentSandboxBridge`-Eigenschaft ermöglicht dem übergeordneten Frame, eine Schnittstelle zum Inhalt im untergeordneten Frame zu öffnen. Um eine Schnittstelle zu öffnen, stellen Sie die `parentSandbox`-Eigenschaft für eine Funktion oder ein Objekt im übergeordneten Frame ein. Sie können dann vom Inhalt im untergeordneten Frame aus auf das Objekt oder die Funktion zugreifen. Im folgenden Beispiel wird veranschaulicht, wie ein Skript, das im übergeordneten Frame ausgeführt wird, ein Objekt, das eine save-Funktion enthält, für einen untergeordneten Frame öffnen kann:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").parentSandboxBridge = interface;
```

Unter Verwendung dieser Schnittstelle könnte Inhalt im untergeordneten Frame Text in einer Datei mit dem Namen „save.txt“ speichern. Der Inhalt hätte jedoch ansonsten keinen Zugriff auf das Dateisystem. Im Allgemeinen sollte Anwendungsinhalt die kleinstmögliche Schnittstelle zu anderen Sandboxen öffnen. Der untergeordnete Inhalt könnte die save-Funktion wie folgt aufrufen:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Wenn untergeordneter Inhalt versucht, eine Eigenschaft des `parentSandboxBridge`-Objekts festzulegen, gibt die Laufzeitumgebung eine SecurityError-Ausnahme aus. Wenn übergeordneter Inhalt versucht, eine Eigenschaft des `childSandboxBridge`-Objekts festzulegen, gibt die Laufzeitumgebung eine SecurityError-Ausnahme aus.

Codebeschränkungen für Inhalt in unterschiedlichen Sandboxen

Adobe AIR 1.0 und höher

Wie am Anfang dieses Abschnitts unter „HTML-Sicherheit in Adobe AIR“ auf Seite 1158 erwähnt, erzwingt die Laufzeitumgebung Regeln und stellt Mechanismen bereit, um mögliche Sicherheitslücken in HTML und JavaScript zu schließen. Im vorliegenden Abschnitt werden diese Beschränkungen beschrieben. Wenn Code versucht, diese beschränkten APIs aufzurufen, gibt die Laufzeitumgebung einen Fehler mit der Meldung „Adobe AIR runtime security violation for JavaScript code in the application security sandbox“ (Adobe AIR-Laufzeitumgebung: Sicherheitsverletzung bei JavaScript-Code in der Sicherheits-Sandbox der Anwendung) aus.

Weitere Informationen finden Sie unter „Vermeiden von sicherheitsbezogenen JavaScript-Fehlern“ auf Seite 1049.

Beschränkungen bei Verwendung der JavaScript-Funktion eval() und ähnlicher Techniken

Adobe AIR 1.0 und höher

Für HTML-Inhalt in der Sicherheits-Sandbox der Anwendung gelten Beschränkungen für die Verwendung von APIs, die Strings dynamisch in ausführbaren Code umwandeln können, nachdem der Code geladen wurde (nachdem das `onload`-Ereignis des `body`-Elements ausgelöst und die Ausführung der `onload`-Prozedurfunktion beendet wurde). Auf diese Weise wird verhindert, dass die Anwendung ungewollt Code aus anderen Quellen als der Anwendung (zum Beispiel von potenziell unsicheren Netzwerkdomänen) einfügt und ausführt.

Wenn Ihre Anwendung zum Beispiel Stringdaten von einer Remote-Quelle verwendet, um in die innerHTML-Eigenschaft eines DOM-Elements zu schreiben, könnte der String ausführbaren (JavaScript-)Code enthalten, der unsichere Operationen ausführen könnte. Während der Inhalt geladen wird, besteht jedoch kein Risiko, dass Remote-Strings in das DOM geschrieben werden.

Eine Beschränkung besteht für die Verwendung der JavaScript-Funktion `eval()`. Nachdem Code in die Anwendungs-Sandbox geladen und die onload-Ereignisprozedur verarbeitet wurde, können Sie die `eval()`-Funktion nur eingeschränkt verwenden. Die folgenden Regeln gelten für die Verwendung der `eval()`-Funktion *nachdem* Code aus der Anwendungs-Sandbox geladen wurde:

- Ausdrücke, die Literale beinhalten, sind zulässig. Zum Beispiel:

  ```
  eval("null");
  eval("3 + .14");
  eval("'foo'");
  ```

- Objektliterale sind zulässig, wie im Folgenden:

  ```
  { prop1: val1, prop2: val2 }
  ```

- Set-/Get-Methoden für Objektliterale sind *unzulässig*, wie im Folgenden:

  ```
  { get prop1() { ... }, set prop1(v) { ... } }
  ```

- Arrayliterale sind zulässig, wie im Folgenden:

  ```
  [ val1, val2, val3 ]
  ```

- Ausdrücke, die Eigenschaften beinhalten, sind *unzulässig*, wie im Folgenden:

  ```
  a.b.c
  ```

- Der Funktionsaufruf ist *unzulässig*.
- Funktionsdefinitionen sind *unzulässig*.
- Das Einstellen von Eigenschaften ist *unzulässig*.
- Funktionsliterale sind *unzulässig*.

Während der Code geladen wird, vor dem `onload`-Ereignis, und während der Ausführung der `onload`-Ereignisprozedurfunktion, gelten diese Beschränkungen für Inhalte in der Sicherheits-Sandbox der Anwendung jedoch nicht.

Nachdem der Code geladen wurde, führt der folgende Code zum Beispiel dazu, dass die Laufzeitumgebung eine Ausnahme ausgibt:

```
eval("alert(44)");
eval("myFunction(44)");
eval("NativeApplication.applicationID");
```

Dynamisch generierter Code, wie zum Beispiel der beim Aufrufen der `eval()`-Funktion generierte Code, würde ein Sicherheitsrisiko darstellen, wenn er innerhalb der Anwendungs-Sandbox zulässig wäre. Eine Anwendung könnte zum Beispiel unbeabsichtigt einen String ausführen, der aus einer Netzwerkdomäne geladen wurde und schädlichen Code enthält. Dieser Code könnte beispielsweise dazu führen, dass Dateien auf dem Computer des Benutzer gelöscht oder verändert werden. Es wäre auch möglich, dass der Code den Inhalt einer lokalen Daten an eine nicht vertrauenswürdige Netzwerkdomäne weitergibt.

Dynamischer Code kann auf folgende Weise generiert werden:

- Aufrufen der `eval()`-Funktion.
- Verwenden von `innerHTML`-Eigenschaften oder DOM-Funktionen, um Skript-Tags einzufügen, die ein Skript außerhalb des Anwendungsverzeichnisses laden.
- Verwenden von `innerHTML`-Eigenschaften oder DOM-Funktionen, um Skript-Tags einzufügen, die Inlinecode enthalten (anstatt ein Skript über das `src`-Attribut zu laden).
- Einstellen des `src`-Attributs für ein `script`-Tag, um eine JavaScript-Datei zu laden, die sich außerhalb des Anwendungsverzeichnisses befindet.
- Verwenden des `javascript`-URL-Schemas (wie in `href="javascript:alert('Test')"`).
- Verwenden der `setInterval()`- oder `setTimout()`-Funktion, wenn der erste Parameter (der die asynchrone Ausführung der Funktion festlegt) ein (zu evaluierender) String ist anstatt ein Funktionsname (wie in `setTimeout('x = 4', 1000)`).
- Aufrufen der `document.write()`- oder `document.writeln()`-Methode.

Code in der Anwendungs-Sandbox kann diese Methoden nur verwenden, während Inhalt geladen wird.

Diese Beschränkungen verhindern *nicht* die Verwendung von `eval()` mit JSON-Objektliteralen. Auf diese Weise kann Ihr Anwendungsinhalt mit der JSON-JavaScript-Bibliothek arbeiten. Sie können jedoch keinen überladenen JSON-Code (mit Ereignisprozeduren) verwenden.

Überprüfen Sie bei anderen Ajax-Frameworks und JavaScript-Codebibliotheken, ob der Code im Framework bzw. in der Bibliothek im Rahmen dieser Beschränkungen für dynamisch generierten Code funktioniert. Ist dies nicht der Fall, platzieren Sie alle Inhalte, die das Framework oder die Bibliothek verwenden, in einer anwendungsfremden Sandbox. Einzelheiten finden Sie unter „Einschränkungen für JavaScript in AIR“ auf Seite 1120 und „Skripterstellung zwischen Anwendungsinhalt und anwendungsfremdem Inhalt“ auf Seite 1170. Adobe unterhält eine Liste von Ajax-Frameworks, die die Anwendungs-Sandbox unterstützen, unter http://www.adobe.com/de/products/air/develop/ajax/features/.

Anders als Inhalt in der Anwendungs-Sandbox *kann* JavaScript-Inhalt in einer anwendungsfremden Sandbox jederzeit die `eval()`-Funktion aufrufen, um dynamisch generierten Code auszuführen.

Beschränkungen für den Zugriff auf AIR-APIs (für anwendungsfremde Sandboxen)

Adobe AIR 1.0 und höher

JavaScript-Code in einer anwendungsfremden Sandbox hat keinen Zugriff auf das `window.runtime`-Objekt und der Code kann keine AIR-APIs ausführen. Wenn Inhalt in einer anwendungsfremden Sandbox den folgenden Code aufruft, gibt die Anwendung eine TypeError-Ausnahme aus:

```
try {
    window.runtime.flash.system.NativeApplication.nativeApplication.exit();
}
catch (e)
{
    alert(e);
}
```

Der Ausnahmetyp ist TypeError (nicht definierter Wert), da Inhalt in der anwendungsfremden Sandbox das `window.runtime`-Objekt nicht erkennt und es deshalb als nicht definierten Wert betrachtet.

Sie können Funktionen der Laufzeitumgebung für Inhalt in einer anwendungsfremden Sandbox öffnen, indem Sie eine Skriptbrücke verwenden. Weitere Informationen finden Sie unter „Skripterstellung zwischen Anwendungsinhalt und anwendungsfremdem Inhalt“ auf Seite 1170.

Beschränkungen bei der Verwendung von XMLHttpRequest-Aufrufen

Adobe AIR 1.0 und höher

HTML-Inhalt in der Anwendungs-Sandbox kann keine synchronen XMLHttpRequest-Methoden verwenden, um Daten von außerhalb der Anwendungs-Sandbox zu laden, während der HTML-Inhalt geladen wird und während ein `onLoad`-Ereignis auftritt.

Standardmäßig ist es für HTML-Inhalt in anwendungsfremden Sandboxen nicht zulässig, mit dem JavaScript-XMLHttpRequest-Objekt Daten aus anderen Domänen als der aufrufenden Domäne zu laden. Ein `frame`- oder `iframe`-Tag kann ein `allowcrosscomainxhr`-Attribut beinhalten. Wenn dieses Attribut auf einen anderen Wert als null eingestellt ist, kann der Inhalt im Frame oder Inlineframe das XMLHttpRequest-Objekt aus JavaScript verwenden, um Daten auch aus Domänen zu laden, bei denen es sich nicht um die Domäne des Codes handelt, der die Anforderung aufruft:

```
<iframe id="UI"
    src="http://example.com/ui.html"
    sandboxRoot="http://example.com/"
    allowcrossDomainxhr="true"
    documentRoot="app:/">
</iframe>
```

Weitere Informationen finden Sie unter „Skripterstellung zwischen Inhalten in unterschiedlichen Domänen“ auf Seite 1165.

Beschränkungen beim Laden von CSS-, frame-, iframe- und img-Elementen (für Inhalte in anwendungsfremden Sandboxen)

Adobe AIR 1.0 und höher

HTML-Inhalte in Remote-Sandboxen (Netzwerk-Sandboxen) können nur CSS-, `frame`-, `iframe`- und `img`-Inhalt aus Remote-Sandboxen (von Netzwerk-URLs) laden.

HTML-Inhalt in „local-with-filesystem“-, „local-with-networking“- oder „local-trusted“-Sandboxen kann nur CSS-, `frame`-, iframe- und img-Inhalte aus lokalen Sandboxen (nicht aus der Anwendungs-Sandbox oder aus Remote-Sandboxen) laden.

Beschränkungen beim Aufrufen der JavaScript-window.open()-Methode

Adobe AIR 1.0 und höher

Wenn ein Fenster, das über einen Aufruf der JavaScript-Methode `window.open()` erstellt wurde, Inhalt aus einer anwendungsfremden Sandbox anzeigt, beginnt der Titel des Fensters mit dem Titel des Hauptfensters (des aufrufenden Fensters), gefolgt von einem Doppelpunkt. Sie können diesen Teil des Fenstertitels nicht mithilfe von Code vom Bildschirm bewegen.

Inhalt in anwendungsfremden Sicherheits-Sandboxen kann die JavaScript-Methode `window.open()` nur als Antwort auf ein Ereignis, das durch eine Benutzerinteraktion mit Maus oder Tastatur ausgelöst wurde, erfolgreich aufrufen. Auf diese Weise wird verhindert, dass anwendungsfremder Inhalt Fenster erstellt, die möglicherweise in betrügerischer Absicht verwendet werden (zum Beispiel für Phishing-Angriffe). Außerdem kann die Ereignisprozedur für Maus- und Tastaturereignisse nicht festlegen, dass die `window.open()`-Methode mit Verzögerung ausgeführt wird (zum Beispiel durch Aufrufen der `setTimeout()`-Funktion).

Inhalt in Remote-Sandboxen (Netzwerk-Sandboxen) kann die `window.open()`-Methode nur verwenden, um Inhalt in Remote-Netzwerk-Sandboxen zu öffnen. Dieser Inhalt kann die `window.open()`-Methode nicht zum Öffnen von Inhalten in der Anwendungs-Sandbox oder in lokalen Sandboxen verwenden.

Inhalt in den Sandboxen „local-with-filesystem“, „local-with-networking“ oder „local-trusted“ (siehe „Sicherheits-Sandboxen“ auf Seite 1117) kann die `window.open()`-Methode nur zum Öffnen von Inhalten in lokalen Sandboxen verwenden. Dieser Inhalt kann die `window.open()`-Methode nicht zum Öffnen von Inhalten in der Anwendungs-Sandbox oder in Remote-Sandboxen verwenden.

Fehler beim Aufrufen von beschränktem Code

Adobe AIR 1.0 und höher

Wenn Sie Code aufrufen, der aufgrund dieser Sicherheitsregeln in einer bestimmten Sandbox nicht verwendet werden darf, gibt die Laufzeitumgebung einen JavaScript-Fehler aus: "Adobe AIR-Laufzeitumgebung: Sicherheitsverletzung bei JavaScript-Code in der Sicherheits-Sandbox der Anwendung."

Weitere Informationen finden Sie unter „Vermeiden von sicherheitsbezogenen JavaScript-Fehlern“ auf Seite 1049.

Sandbox-Schutz beim Laden von HTML-Inhalten aus einem String

Adobe AIR 1.0 und höher

Mit der `loadString()`-Methode der HTMLLoader-Klasse können Sie HTML-Inhalt zur Laufzeit erstellen. Daten, die Sie als HTML-Inhalt verwenden, können jedoch schadhaft sein, wenn die Daten aus einer unsicheren Internetquelle geladen werden. Aus diesem Grund wird mit der `loadString()`-Methode erstellter HTML-Inhalt standardmäßig nicht in der Anwendungs-Sandbox abgelegt und hat keinen Zugriff auf AIR-APIs. Sie können jedoch die `placeLoadStringContentInApplicationSandbox`-Eigenschaft eines HTMLLoader-Objekts auf „true“ einstellen, um mit der `loadString()`-Methode erstellten HTML-Inhalt in der Anwendungs-Sandbox zu platzieren. Weitere Informationen finden Sie unter „Laden von HTML-Inhalten aus einem String“ auf Seite 1048.

Skripterstellung zwischen Inhalten in unterschiedlichen Domänen

Adobe AIR 1.0 und höher

AIR-Anwendungen erhalten bei der Installation besondere Berechtigungen. Es ist wichtig, dass diese Berechtigungen nicht auf andere Inhalte übergehen, darunter Remote-Dateien und lokale Dateien, die nicht Teil der Anwendung sind.

Die AIR-Sandboxbrücke

Adobe AIR 1.0 und höher

Normalerweise kann Inhalt aus anderen Domänen keine Skripts in anderen Domänen aufrufen. Um AIR-Anwendungen vor der versehentlichen Preisgabe von Informationen oder Kontrolle zu schützen, gelten die folgenden Beschränkungen für Inhalte in der Sicherheits-Sandbox `application` (Inhalte, die mit der Anwendung installiert werden):

- Code in der Anwendungs-Sandbox kann nicht durch Aufrufen der `Security.allowDomain()`-Methode mit anderen Sandboxen interagieren. Wenn diese Methode aus der Anwendungs-Sandbox aufgerufen wird, wird ein Fehler ausgegeben.
- Das Importieren von anwendungsfremden Inhalten in die Anwendungs-Sandbox durch Festlegen der `LoaderContext.securityDomain`-Eigenschaft oder der `LoaderContext.applicationDomain`-Eigenschaft wird verhindert.

In einigen Fällen ist es jedoch erforderlich, dass die AIR-Hauptanwendung Inhalten aus einer Remote-Domäne den kontrollierten Zugriff auf Skripts in der AIR-Hauptanwendung ermöglicht oder umgekehrt. Zu diesem Zweck gibt es in der Laufzeitumgebung so genannte *Sandboxbrücken*, die als Schnittstelle zwischen zwei Sandboxen fungieren. Eine Sandboxbrücke kann die explizite Interaktion zwischen Remote- und Anwendungs-Sandboxen ermöglichen.

Die Sandboxbrücke öffnet zwei Objekte, auf die sowohl geladene als auch ladende Skripts zugreifen können:

- Das `parentSandboxBridge`-Objekt ermöglicht, dass ladender Inhalt Eigenschaften und Funktionen für Skripts im geladenen Inhalt öffnet.
- Das `childSandboxBridge`-Objekt ermöglicht, dass geladener Inhalt Eigenschaften und Funktionen für Skripts im ladenden Inhalt öffnet.

Objekte, die über die Sandboxbrücke geöffnet werden, werden durch Werte, nicht durch Verweise übergeben. Alle Daten werden serialisiert. Dies bedeutet, dass die von einer Seite der Brücke geöffneten Objekte nicht von der anderen Seite eingestellt werden können und dass geöffnete Objekte nicht typisiert sind. Des Weiteren lassen sich nur einfache Objekte und Funktionen öffnen; Sie können keine komplexen Objekte öffnen.

Wenn untergeordneter Inhalt versucht, eine Eigenschaft des parentSandboxBridge-Objekts festzulegen, gibt die Laufzeitumgebung eine SecurityError-Ausnahme aus. Wenn übergeordneter Inhalt versucht, eine Eigenschaft des childSandboxBridge-Objekts festzulegen, gibt die Laufzeitumgebung eine SecurityError-Ausnahme aus.

Beispiel für eine Sandboxbrücke (SWF)

Adobe AIR 1.0 und höher

In einer Musikshop-AIR-Anwendung sollen Remote-SWF-Dateien den Preis der Alben übermitteln, dabei sollen sie aber nicht angeben, ob es sich bei dem Preis um ein Sonderangebot handelt. Zu diesem Zweck stellt eine StoreAPI-Klasse eine Methode bereit, um den Preis anzugeben, der Angebotspreis wird jedoch verborgen. Eine Instanz dieser StoreAPI-Klasse wird der `parentSandboxBridge`-Eigenschaft des LoaderInfo-Objekts des Loader-Objekts, das die Remote-SWF-Datei lädt, zugewiesen.

Der Code für den AIR-Musikshop sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
title="Music Store" creationComplete="initApp()">
    <mx:Script>
        import flash.display.Loader;
        import flash.net.URLRequest;

        private var child:Loader;
        private var isSale:Boolean = false;

        private function initApp():void {
            var request:URLRequest =
                    new URLRequest("http://[www.yourdomain.com]/PriceQuoter.swf")

            child = new Loader();
            child.contentLoaderInfo.parentSandboxBridge = new StoreAPI(this);
            child.load(request);
            container.addChild(child);
        }
        public function getRegularAlbumPrice():String {
            return "$11.99";
        }
        public function getSaleAlbumPrice():String {
            return "$9.99";
        }
        public function getAlbumPrice():String {
            if(isSale) {
                return getSaleAlbumPrice();
            }
            else {
                return getRegularAlbumPrice();
            }
        }
    </mx:Script>
    <mx:UIComponent id="container" />
</mx:WindowedApplication>
```

Das StoreAPI-Objekt ruft die Hauptanwendung auf, um den normalen Albumpreis abzurufen, gibt jedoch „Not available“ (Nicht verfügbar) zurück, wenn die `getSaleAlbumPrice()`-Methode aufgerufen wird. Mit dem folgenden Code wird die StoreAPI-Klasse definiert:

```
public class StoreAPI
{
    private static var musicStore:Object;

    public function StoreAPI(musicStore:Object)
    {
        this.musicStore = musicStore;
    }

    public function getRegularAlbumPrice():String {
        return musicStore.getRegularAlbumPrice();
    }

    public function getSaleAlbumPrice():String {
        return "Not available";
    }

    public function getAlbumPrice():String {
        return musicStore.getRegularAlbumPrice();
    }
}
```

Der folgende Code stellt ein Beispiel einer PriceQuoter-SWF-Datei dar, die den Preis meldet, den Angebotspreis jedoch nicht melden kann:

```
package
{
    import flash.display.Sprite;
    import flash.system.Security;
    import flash.text.*;

    public class PriceQuoter extends Sprite
    {
        private var storeRequester:Object;

        public function PriceQuoter() {
            trace("Initializing child SWF");
            trace("Child sandbox: " + Security.sandboxType);
            storeRequester = loaderInfo.parentSandboxBridge;

            var tf:TextField = new TextField();
            tf.autoSize = TextFieldAutoSize.LEFT;
            addChild(tf);

            tf.appendText("Store price of album is: " + storeRequester.getAlbumPrice());
            tf.appendText("\n");
            tf.appendText("Sale price of album is: " + storeRequester.getSaleAlbumPrice());
        }
    }
}
```

Beispiel für eine Sandboxbrücke (HTML)

Adobe AIR 1.0 und höher

Bei HTML-Inhalt werden die `parentSandboxBridge`-Eigenschaft und die `childSandboxBridge`-Eigenschaft dem JavaScript-window-Objekt eines untergeordneten Dokuments hinzugefügt. Ein Beispiel für das Einrichten von Brückenfunktionen in HTML-Inhalt finden Sie unter „Einrichten einer Schnittstelle für Sandbox-Brücken“ auf Seite 1068.

Begrenzen der API-Öffnung

Adobe AIR 1.0 und höher

Beim Öffnen von Sandboxbrücken ist es wichtig, APIs auf hoher Ebene zu öffnen, die den Grad der missbräuchlichen Nutzung einschränken. Beachten Sie, dass der Inhalt, der Ihre Brückenimplementierung aufruft, schädlich sein kann (zum Beispiel durch Einschleusen von Code). Das Öffnen einer `readFile(path:String)`-Methode (die den Inhalt einer zweifelhaften Datei liest) über die Brücke stellt zum Beispiel ein Risiko dar. Es wäre besser, eine `readApplicationSetting()`-API zu öffnen, die nicht den Pfad übernimmt und eine bestimmte Datei liest. Der semantischere Ansatz begrenzt den Schaden, den eine Anwendung anrichten kann, wenn ein Teil von ihr schädlich ist.

Verwandte Hilfethemen

„Cross-Scripting von Inhalten in unterschiedlichen Sicherheits-Sandboxen“ auf Seite 1067

„Die AIR-Anwendungs-Sandbox“ auf Seite 1118

Schreiben auf eine Festplatte

Adobe AIR 1.0 und höher

Anwendungen, die in einem Webbrowser ausgeführt werden, interagieren nur begrenzt mit dem lokalen Dateisystem des Benutzers. Webbrowser implementieren Sicherheitsrichtlinien, die dafür sorgen, dass der Computer des Benutzers nicht durch das Laden von Webinhalten gefährdet wird. SFW-Dateien, die über Flash Player in einem Browser ausgeführt werden, können zum Beispiel nicht direkt mit Dateien interagieren, die sich bereits auf dem Computer des Benutzers befinden. Gemeinsam genutzte Objekte und Cookies können auf den Computer des Benutzers geschrieben werden, um Benutzereinstellungen und andere Daten zu erhalten, weitere Interaktionen mit dem Dateisystem sind jedoch nicht möglich. Da AIR-Anwendungen nativ installiert werden, gilt für sie ein anderer Sicherheitsvertrag, der die Möglichkeit einschließt, im lokalen Dateisystem zu lesen und zu schreiben.

Dies stellt für Entwickler eine große Verantwortung dar. Unbeabsichtigte Sicherheitslücken der Anwendung gefährden nicht nur die Funktionalität der Anwendung, sondern auch die Integrität des Benutzercomputers. Aus diesem Grund sollten Entwickler den Abschnitt „Empfohlene Sicherheitsverfahren für Entwickler“ auf Seite 1171 lesen.

AIR-Entwickler können auf Dateien im lokalen Dateisystem zugreifen bzw. in diese schreiben, indem verschiedene URL-Schemakonventionen verwendet werden:

URL-Schema	Beschreibung
app:/	Ein Alias für das Anwendungsverzeichnis. Dateien, auf die von diesem Pfad zugegriffen wird, werden der Anwendungs-Sandbox zugewiesen und verfügen über alle von der Laufzeitumgebung gewährten Berechtigungen.
app-storage:/	Ein Alias für das lokale Speicherverzeichnis, von der Laufzeitumgebung standardisiert. Dateien, auf die von diesem Pfad zugegriffen wird, werden einer anwendungsfremden Sandbox zugewiesen.
file:///	Ein Alias, der das Stammverzeichnis der Festplatte des Benutzers darstellt. Eine Datei, auf die von diesem Pfad zugegriffen wird, wird einer Anwendungs-Sandbox zugewiesen, falls die Datei im Anwendungsverzeichnis vorhanden ist, andernfalls einer anwendungsfremden Sandbox.

Hinweis: *AIR-Anwendungen können mit dem app-URL-Schema nicht den Inhalt verändern. Des Weiteren kann das Anwendungsverzeichnis aufgrund von Administratoreinstellungen schreibgeschützt sein.*

Falls keine Administratoreinschränkungen für den Computer des Benutzers festgelegt wurden, verfügen AIR-Anwendungen über die Berechtigung, in jeden Speicherort auf der Festplatte des Benutzers zu schreiben. Entwicklern wird empfohlen, den `app-storage:/`-Pfad als lokalen Speicher für anwendungsbezogene Speichervorgänge zu verwenden. Dateien, die unter `app-storage:/` von einer Anwendung geschrieben werden, werden an einem Standardspeicherort abgelegt:

- Mac OS: Das Speicherverzeichnis einer Anwendung ist `<appData>/<appId>/Local Store/`, wobei `<appData>` der Einstellungsordner des Benutzers ist. Normalerweise ist dies `/<Benutzer>/>/Library//Voreinstellungen`
- Windows: Das Speicherverzeichnis einer Anwendung ist `<appData>\<appId>\Local Store\`, wobei `<appData>` de spezielle CSIDL_APPDATA-Ordner des Benutzers ist. Normalweise ist dies `C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten`
- Linux: `<appData>/<appID>/Local Store/`wobei `<appData>/home/<Benutzer>/.appdata` ist

Wenn eine Anwendung mit den vorhandenen Dateien im Dateisystem des Benutzers interagieren soll, lesen Sie unbedingt den Abschnitt „Empfohlene Sicherheitsverfahren für Entwickler“ auf Seite 1171.

Sicheres Arbeiten mit nicht vertrauenswürdigen Inhalten

Adobe AIR 1.0 und höher

Inhalt, der nicht der Anwendungs-Sandbox zugewiesen ist, kann zusätzliche Scriptingfunktionen für Ihre Anwendung bereitstellen, jedoch nur, wenn die Sicherheitsanforderungen der Laufzeitumgebung erfüllt sind. In diesem Abschnitt wird der AIR-Sicherheitsvertrag mit anwendungsfremden Inhalten beschrieben.

Security.allowDomain()

Adobe AIR 1.0 und höher

AIR-Anwendungen beschränken den Scriptingzugriff für anwendungsfremden Inhalt stärker, als das Flash Player -Browser-Plug-In den Scriptingzugriff für nicht vertrauenswürdigen Inhalt beschränkt. Wenn bei Flash Player im Browser eine SWF-Datei, die der `local-trusted`-Sandbox zugewiesen ist, die `System.allowDomain()`-Methode aufruft, wird der Scriptingzugriff jeder SWF-Datei aus der angegebenen Domäne gewährt. Die entsprechende Vorgehensweise ist für `application`-Inhalt in AIR-Anwendungen nicht zulässig, da auf diese Weise der riskante Zugriff auf anwendungsfremde Dateien im Dateisystem des Benutzers möglich wäre. Remote-Dateien können nicht direkt auf die Anwendungs-Sandbox zugreifen, unabhängig von Aufrufen der `Security.allowDomain()`-Methode.

Skripterstellung zwischen Anwendungsinhalt und anwendungsfremdem Inhalt

Adobe AIR 1.0 und höher

Für Anwendungen, die Skripts zwischen Anwendungsinhalt und anwendungsfremdem Inhalt erstellen, gelten komplexere Sicherheitsregeln. Dateien, die sich nicht in der Anwendungs-Sandbox befinden, können nur über eine Sandboxbrücke auf die Eigenschaften und Methoden von Dateien in der Anwendungs-Sandbox zugreifen. Sandboxbrücken fungieren als Schnittstelle zwischen Anwendungsinhalt und anwendungsfremdem Inhalt und ermöglichen so die explizite Interaktion zwischen den beiden Dateien. Bei richtiger Verwendung bieten Sandboxbrücken zusätzliche Sicherheit, da anwendungsfremder Inhalt nicht auf Objektverweise zugreifen kann, die Teil des Anwendungsinhalts sind.

Der Vorteil von Sandboxbrücken lässt sich am besten anhand eines Beispiels veranschaulichen. Angenommen, ein AIR-Musikshop möchte Werbekunden, die ihre eigenen SWF-Dateien erstellen möchten, eine API zur Verfügung stellen, mit der die Shop-Anwendung dann kommunizieren kann. Der Shop möchte Werbekunden die Möglichkeit geben, Künstler und CDs im Shop nachzuschlagen, möchte aus Sicherheitsgründen aber auch einige Methoden und Eigenschaften aus der SWF-Datei des Drittanbieters isolieren.

Diese Funktionalität kann durch eine Sandboxbrücke erreicht werden. Standardmäßig hat Inhalt, der extern in eine AIR-Anwendung geladen wird, keinen Zugriff auf die Methoden und Eigenschaften in der Hauptanwendung. Mit einer benutzerdefinierten Sandboxbrückenimplementierung kann ein Entwickler Remote-Inhalten Dienste anbieten, ohne diese Methoden oder Eigenschaften zu öffnen. Sie können sich die Sandboxbrücke als Tor zwischen vertrauenswürdigem und nicht vertrauenswürdigem Inhalt vorstellen, das die Kommunikation zwischen ladendem und geladenem Inhalt ohne Preisgabe der Objektverweise ermöglicht.

Weitere Informationen zur sicheren Verwendung von Sandboxbrücken finden Sie unter „Skripterstellung zwischen Inhalten in unterschiedlichen Domänen“ auf Seite 1165.

Schutz der dynamischen Generierung von unsicheren SWF-Inhalten

Adobe AIR 1.0 und höher

Die `Loader.loadBytes()`-Methode gibt einer Anwendung die Möglichkeit, SWF-Inhalte aus einem Byte-Array zu generieren. Einschleusungsangriffe mit Daten, die von Remote-Quellen geladen werden, können beim Laden von Inhalten jedoch schweren Schaden anrichten. Dies gilt besonders dann, wenn Daten in die Anwendungs-Sandbox geladen werden, wo der generierte SWF-Inhalt auf sämtliche AIR-APIs zugreifen kann.

Es gibt legitime Einsatzmöglichkeiten der `loadBytes()`-Methode, ohne ausführbaren SWF-Code zu generieren. Mit der `loadBytes()`-Methode können Sie zum Beispiel Bilddaten generieren, um das Timing der Bildanzeige zu steuern. Es gibt jedoch auch legitime Verwendungszwecke, die ausführbaren Code benötigen, zum Beispiel die dynamische Erstellung von SWF-Inhalten für die Audiowiedergabe. In AIR lässt die `loadBytes()`-Methode das Laden von SWF-Inhalten standardmäßig *nicht* zu; Sie können nur Bildinhalte laden. In AIR verfügt die `loaderContext`-Eigenschaft der `loadBytes()`-Methode über die `allowLoadBytesCodeExecution`-Eigenschaft, die Sie explizit auf `true` setzen können, um zuzulassen, dass die Anwendung `loadBytes()` zum Laden von ausführbarem SWF-Inhalt verwendet. Im folgenden Code wird die Verwendung dieser Funktion veranschaulicht:

```
var loader:Loader = new Loader();
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.allowLoadBytesCodeExecution = true;
loader.loadBytes(bytes, loaderContext);
```

Wenn Sie `loadBytes()` aufrufen, um SWF-Inhalt zu laden, und die `allowLoadBytesCodeExecution`-Eigenschaft des LoaderContext-Objekts ist auf `false` (die Standardeinstellung) eingestellt, gibt das Loader-Objekt eine SecurityError-Ausnahme aus.

Hinweis: *In einer zukünftigen Version von Adobe AIR wird diese API möglicherweise geändert. In diesem Fall müssen Sie ggf. Inhalt, der die* `allowLoadBytesCodeExecution`*-Eigenschaft der LoaderContext-Klasse verwendet, neu kompilieren.*

Empfohlene Sicherheitsverfahren für Entwickler

Adobe AIR 1.0 und höher

Obwohl AIR-Anwendungen mithilfe von Webtechnologien erstellt werden, ist es für Entwickler wichtig, zu beachten, dass sie nicht innerhalb der Sicherheits-Sandbox des Browsers arbeiten. Das bedeutet, dass es möglich ist, AIR-Anwendungen zu erstellen, die das lokale System schädigen können, sei es absichtlich oder versehentlich. AIR versucht, dieses Risiko zu minimieren, es gibt jedoch immer noch Möglichkeiten, Sicherheitsverletzungen herbeizuführen. In diesem Abschnitt werden wichtige potenzielle Sicherheitslücken behandelt.

Risiken beim Importieren von Dateien in die Anwendungs-Sandbox

Adobe AIR 1.0 und höher

Dateien, die sich im Anwendungsverzeichnis befinden, werden der Anwendungs-Sandbox zugewiesen und verfügen über sämtliche Berechtigungen der Laufzeitumgebung. Anwendungen, die in das lokale Dateisystem schreiben, sollten in `app-storage:/` schreiben. Dieses Verzeichnis besteht separat von den Anwendungsdateien auf dem Computer des Benutzer, daher werden die Dateien nicht der Anwendungs-Sandbox zugewiesen und stellen ein vermindertes Sicherheitsrisiko dar. Entwicklern wird empfohlen, Folgendes in Betracht zu ziehen:

- Schließen Sie eine Datei nur dann in eine AIR-Datei (in die installierte Anwendung) ein, wenn dies wirklich erforderlich ist.
- Schließen Sie eine Skripterstellungsdatei nur dann in die AIR-Datei (in die installierte Anwendung) ein, wenn deren Verhalten vollkommen klar und vertrauenswürdig ist.
- Schreiben Sie keinen Inhalt in das Anwendungsverzeichnis und verändern Sie keinen dort abgelegten Inhalt. Die Laufzeitumgebung verhindert, dass Anwendungen mithilfe des `app:/`-URL-Schemas Dateien und Verzeichnisse schreiben oder ändern, indem eine SecurityError-Ausnahme ausgegeben wird.
- Verwenden Sie keine Daten aus einer Netzwerkquelle als Parameter für Methoden der AIR-API, die möglicherweise die Codeausführung zur Folge haben. Dazu gehört auch die Verwendung der `Loader.loadBytes()`-Methode und der JavaScript-Funktion `eval()`.

Risiken bei der Verwendung einer externen Quelle zur Pfadbestimmung

Adobe AIR 1.0 und höher

Eine AIR-Anwendung kann geschädigt werden, wenn externe Daten oder Inhalte verwendet werden. Gehen Sie aus diesem Grund mit größter Vorsicht vor, wenn Sie Daten aus dem Netzwerk oder Dateisystem verwenden. Die Verantwortung für die Vertrauenswürdigkeit liegt letztendlich auf Seiten der Entwickler und der Netzwerkverbindungen, die sie herstellen; das Laden fremder Daten stellt aber immer ein Risiko dar und sollte nicht für die Eingabe bei kritischen Operationen verwendet werden. Entwicklern wird von Folgendem abgeraten:

- Bestimmen von Dateinamen mithilfe von Daten aus einer Netzwerkquelle
- Konstruieren einer URL, mit der die Anwendung persönliche Angeben sendet, mithilfe von Daten aus einer Netzwerkquelle

Risiken beim Verwenden, Speichern oder Übertragen von unsicheren Benutzerangaben

Adobe AIR 1.0 und höher

Das Speichern von Benutzerangaben im lokalen Dateisystem des Benutzers beinhaltet das Risiko, dass diese Angaben missbräuchlich genutzt oder beschädigt werden. Entwicklern wird empfohlen, Folgendes in Betracht zu ziehen:

- Wenn Benutzerangaben lokal gespeichert werden müssen, verschlüsseln Sie die Angaben beim Schreiben in das lokale Dateisystem. Die Laufzeitumgebung stellt über die EncryptedLocalStore-Klasse eine verschlüsselte Speicherung zur Verfügung, die für jede installierte Anwendung eindeutig ist. Weitere Informationen finden Sie unter „Verschlüsselter lokaler Speicher“ auf Seite 757.
- Senden Sie unverschlüsselte Benutzeranmeldedaten nur dann an eine Netzwerkquelle, wenn Sie sicher sind, dass die Quelle vertrauenswürdig ist und wenn das sichere Protokoll HTTPS oder Transport Layer Security (TLS) verwendet wird.
- Geben Sie bei der Erstellung von Benutzerangaben nie ein Standardkennwort an, sondern lassen Sie den Benutzer ein eigenes einrichten. Andernfalls könnten Angreifer, denen das Standardkennwort bekannt ist, alle Angaben des Benutzers erfahren, wenn dieser das Standardkennwort nicht ändert.

Risiken durch Downgrade-Angriffe

Adobe AIR 1.0 und höher

Bei der Anwendungsinstallation überprüft die Laufzeitumgebung, ob bereits eine Version der Anwendung installiert ist. Wenn eine Anwendung bereits installiert ist, vergleicht die Laufzeitumgebung den Versionsstring mit der Version, die installiert werden soll. Unterscheiden sich die Strings, hat der Benutzer die Möglichkeit, seine Installation zu aktualisieren. Die Laufzeitumgebung kann nicht garantieren, dass die neu installierte Version tatsächlich neuer als die alte Version ist, sondern lediglich, dass sie anders ist. Ein Angreifer könnte eine ältere Version an den Benutzer verteilen, um eine Sicherheitslücke auszunutzen. Aus diesem Grund wird Entwicklern geraten, beim Ausführen der Anwendung Versionsüberprüfungen vorzunehmen. Es empfiehlt sich, Anwendungen regelmäßig nach erforderlichen Updates im Netzwerk suchen zu lassen. Selbst wenn ein Angreifer erreicht, dass ein Benutzer eine ältere Version ausführt, kann diese ältere Version dann erkennen, dass sie aktualisiert werden muss. Ein übersichtliches Versionsschema für Ihre Anwendung erschwert es Angreifern, Benutzer zum Installieren einer älteren Version zu bringen.

Codesignaturen

Adobe AIR 1.0 und höher

Alle AIR-Installationsprogrammdateien müssen mit einer Codesignatur versehen sein. Das Signieren von Code ist ein kryptografischer Prozess, um zu bestätigen, dass die angegebene Herkunft der Software korrekt ist. AIR-Anwendungen können entweder mit einem Zertifikat einer externen Zertifizierungsstelle oder durch ein von Ihnen erstelltes selbst unterzeichnetes Zertifikat signiert werden. Es wird dringend empfohlen, ein kommerzielles Zertifikat von einer bekannten Zertifizierungsstelle zu verwenden, um Ihren Benutzern die Gewissheit zu geben, dass sie Ihre Anwendung und keine Fälschung installieren. Selbst unterzeichnete Zertifikate lassen sich jedoch mit `adt` aus dem SDK oder mithilfe von Flash, Flash Builder oder einer anderen Anwendung erstellen, die `adt` für die Zertifikatgenerierung verwendet. Selbst unterzeichnete Zertifikate gewährleisten nicht die Echtheit der installierten Anwendung; deshalb sollten sie nur zum Testen einer Anwendung vor der allgemeinen Veröffentlichung verwendet werden.

Sicherheit auf Android-Geräten

Adobe AIR 2.5 und höher

Wie bei allen Computergeräten erfüllt AIR auch unter Android die Regeln des nativen Sicherheitsmodells. Gleichzeitig gelten in AIR jedoch auch eigene Sicherheitsregeln, die Entwicklern dabei helfen, sichere Anwendungen mit Internetzugang zu erstellen.

Da AIR-Anwendungen unter Android das Android-Paketformat verwenden, fällt die Installation unter das Android-Sicherheitsmodell. Das AIR-Anwendungsinstallationsprogramm wird nicht verwendet.

Das Android-Sicherheitsmodell umfasst drei Hauptaspekte:

- Berechtigungen
- Anwendungssignaturen
- Benutzer-IDs für Anwendungen

Android-Berechtigungen

Zahlreiche Android-Funktionen werden vom Berechtigungsmechanismus des Betriebssystems geschützt. Zur Verwendung einer geschützten Funktion muss der AIR-Anwendungsdeskriptor deklarieren, dass die Anwendung die entsprechenden Berechtigungen benötigt. Wenn ein Benutzer versucht, die Anwendung zu installieren, zeigt das Android-Betriebssystem alle erforderlichen Berechtigungen an, bevor die Installation fortgesetzt wird.

Bei den meisten AIR-Anwendungen müssen die Android-Berechtigungen im Anwendungsdeskriptor angegeben werden. Standardmäßig sind keine Berechtigungen enthalten. Die folgenden Berechtigungen sind für geschützte Android-Funktionen erforderlich, die über die AIR-Laufzeitumgebung verfügbar sind:

ACCESS_COARSE_LOCATION Ermöglicht der Anwendung den Zugriff auf WLAN- und Mobilfunk-Standortdaten über die Geolocation-Klasse.

ACCESS_FINE_LOCATION Ermöglicht der Anwendung den Zugriff auf GPS-Daten über die Geolocation-Klasse.

ACCESS_NETWORK_STATE und ACCESS_WIFI_STATE Ermöglicht der Anwendung den Zugriff auf Netzwerkinformationen über die NetworkInfo-Klasse.

CAMERA Ermöglicht der Anwendung den Zugriff auf die Kamera.

INTERNET Ermöglicht der Anwendung, Netzwerkanfragen zu senden. Ermöglicht auch das Remote-Debugging.

READ_PHONE_STATE Ermöglicht der AIR-Laufzeitumgebung, den Ton stummzuschalten, wenn ein Anruf eingeht.

RECORD_AUDIO Ermöglicht der Anwendung den Zugriff auf das Mikrofon.

WAKE_LOCK und DISABLE_KEYGUARD Ermöglicht der Anwendung, zu verhindern, dass das Gerät in den Standbymodus wechselt, indem die Einstellungen der SystemIdleMode-Klasse verwendet werden.

WRITE_EXTERNAL_STORAGE Ermöglicht der Anwendung, auf die externe Speicherkarte des Geräts zu schreiben.

Anwendungssignaturen

Alle Anwendungspakete, die für die Android-Plattform erstellt werden, müssen signiert sein. Da AIR-Anwendungspakete unter Android im nativen APK-Format von Android erstellt werden, entsprechen die Signaturen den Android-Konventionen und nicht den AIR-Konventionen. Das Signieren von Code unter Android und AIR weist zwar Ähnlichkeiten, aber auch deutliche Unterschiede auf:

- Unter Android überprüft die Signatur, dass der private Schlüssel sich im Besitz des Entwicklers befindet, er überprüft jedoch nicht die Identität des Entwicklers.

- Bei Anwendungen, die über Android Market angeboten werden sollen, muss das Zertifikat mindestens 25 Jahre gültig sein.
- Das Migrieren der Paketsignatur zu einem anderen Zertifikat wird unter Android nicht unterstützt. Wenn ein Update mit einem anderen Zertifikat signiert ist, muss die ursprüngliche Anwendung deinstalliert werden, bevor die aktualisierte Anwendung installiert werden kann.
- Zwei Anwendungen, die mit demselben Zertifikat signiert sind, können eine gemeinsame ID angeben, sodass sie auf den Cache und die Datendateien der jeweils anderen Anwendung zugreifen können. (Eine solche gemeinsame Nutzung ist in AIR nicht möglich.)

Benutzer-IDs für Anwendungen

Android nutzt einen Linux-Kernel. Jeder installierten Anwendung wird eine Benutzer-ID in Linux-Form zugewiesen, die ihre Berechtigungen für verschiedene Vorgänge festlegt, beispielsweise für den Zugriff auf Dateien. Dateien in der Anwendung, im Anwendungsspeicher und in temporären Verzeichnissen sind durch Dateisystemberechtigungen vor dem Zugriff geschützt. Dateien, die auf ein externes Speichermedium (also die SD-Karte) geschrieben werden, können von anderen Anwendungen oder vom Benutzer gelesen, geändert und gelöscht werden, wenn die SD-Karte als Massenspeichergerät auf einem Computer gemountet wird.

Bei Internetanforderungen erhaltene Cookies werden von AIR-Anwendungen nicht gemeinsam genutzt.

Datenschutz bei Verwendung von Hintergrundbildern

Wenn ein Benutzer die Anwendung in den Hintergrund verschiebt, erfassen manche Android-Versionen ein Bildschirmfoto, das als Miniaturansicht in der Liste der zuletzt verwendeten Anwendungen verwendet wird. Dieses Bildschirmfoto wird im Gerätespeicher des Geräts abgelegt und kann von Angreifern aufgerufen werden, die physischen Zugriff auf das Gerät haben.

Wenn Ihre Anwendung vertrauliche Informationen zeigt, sollten Sie darauf achten, dass diese Informationen nicht für das Bildschirmfoto erfasst werden. Das `deactivate`-Ereignis, das vom NativeApplication-Objekt ausgelöst wird, weist darauf hin, dass eine Anwendung nun in den Hintergrund geschaltet wird. Mithilfe dieses Ereignisses können Sie vertrauliche Informationen löschen oder ausblenden.

Verwandte Hilfethemen

Android: Sicherheit und Berechtigungen

Verschlüsselte Daten unter Android

AIR-Anwendungen unter Android können die Verschlüsselungsoptionen der integrierten SQL-Datenbank verwenden, um verschlüsselte Daten zu speichern. Um für optimale Sicherheit zu sorgen, sollte der Verschlüsselungsschlüssel auf einem vom Benutzer eingegebenen Kennwort basieren, das bei jedem Ausführen der Anwendung eingegeben werden muss. Lokal gespeicherte Verschlüsselungsschlüssel oder Kennwörter lassen sich nur schwer oder gar nicht vor einem Angreifer verbergen, der Zugriff auf die Anwendungsdateien hat. Wenn der Eindringling den Schlüssel abrufen kann, bietet das Verschlüsseln der Daten keinen zusätzlichen Schutz. Die Daten sind dann nur durch die vom Android-Betriebssystem bereitgestellte Dateisystemsicherheit geschützt, die auf der Benutzerkennung basiert.

Die EncryptedLocalStore-Klasse kann zum Speichern von Daten verwendet werden, diese Daten werden auf Android-Geräten jedoch nicht verschlüsselt. Das Android-Sicherheitsmodell schützt die Daten mithilfe der Anwendungsbenutzer-ID. Anwendungen, die eine gemeinsame Benutzer-ID verwenden und mit demselben Codesignaturzertifikat signiert werden, nutzen denselben verschlüsselten lokalen Speicher.

Wichtig: *Bei einem gerooteten Smartphone kann jede Anwendung mit Root-Berechtigung auf die Daten aller anderen Anwendungen zugreifen. Deshalb sind im verschlüsselten lokalen Speicher gespeicherte Daten auf einem gerooteten Gerät nicht sicher.*

Sicherheit auf iOS-Geräten

Unter iOS entspricht AIR dem nativen Sicherheitsmodell. Gleichzeitig gelten in AIR jedoch auch eigene Sicherheitsregeln, die Entwicklern dabei helfen, sichere Anwendungen mit Internetzugang zu erstellen.

Da AIR-Anwendungen unter iOS das iOS-Paketformat verwenden, fällt die Installation unter das iOS-Sicherheitsmodell. Das AIR-Anwendungsinstallationsprogramm wird nicht verwendet. Außerdem wird auf iOS-Geräten keine separate AIR-Laufzeit verwendet. Jede AIR-Anwendung enthält den gesamten Code, der für eine korrekte Funktionsweise erforderlich ist.

Anwendungssignaturen

Alle Anwendungspakete, die für die iOS-Plattform erstellt werden, müssen signiert sein. Da für AIR-Anwendungen unter iOS das native iOS IPA-Paketformat verwendet wird, werden sie entsprechend den iOS-Anforderungen signiert, nicht gemäß den AIR-Anforderungen. Das Signieren von Code unter iOS und AIR weist zwar Ähnlichkeiten, aber auch deutliche Unterschiede auf:

- Unter iOS muss das Zertifikat, mit dem eine Anwendung signiert wird, von Apple stammen; Zertifikate von anderen Zertifizierungsstellen können nicht verwendet werden.
- Unter iOS sind von Apple ausgestellte Distributionszertifikate in der Regel ein Jahr lang gültig.

Datenschutz bei Verwendung von Hintergrundbildern

Wenn unter iOS eine Anwendung in den Hintergrund geschaltet wird, erfasst das Betriebssystem ein Bildschirmfoto, mit dem der Übergang animiert wird. Dieses Bildschirmfoto wird im Gerätespeicher des Geräts abgelegt und kann von Angreifern aufgerufen werden, die physischen Zugriff auf das Gerät haben.

Wenn Ihre Anwendung vertrauliche Informationen zeigt, sollten Sie darauf achten, dass diese Informationen nicht für das Bildschirmfoto erfasst werden. Das `deactivate`-Ereignis, das vom NativeApplication-Objekt ausgelöst wird, weist darauf hin, dass eine Anwendung nun in den Hintergrund geschaltet wird. Mithilfe dieses Ereignisses können Sie vertrauliche Informationen löschen oder ausblenden.

Kapitel 65: Verwendung der ActionScript-Beispiele

Das Ausführen eines Codebeispiels für ActionScript 3.0 ist eine der besten Möglichkeiten, die Funktionsweise bestimmter Klassen und Methoden kennen zu lernen. Je nach Gerät, das Sie benutzen oder als Ziel Ihrer Anwendung festgelegt haben, können Sie die Beispiele auf verschiedene Weise verwenden.

Computer mit Flash Professional oder Flash Builder In den Abschnitten „Ausführen von ActionScript 3.0-Beispielen in Flash Professional“ auf Seite 1178 und „Ausführen von ActionScript 3.0-Beispielen in Flash Builder“ auf Seite 1179 finden Sie Informationen zum Ausführen von ActionScript 3.0-Beispielen in diesen Entwicklungsumgebungen. Mit Trace-Anweisungen und anderen Debugging-Tools können Sie Ihr Verständnis der Funktionsweise der Codebeispiele vertiefen.

Mobile Geräte Sie können die ActionScript 3.0-Codebeispiele auf Mobilgeräten ausführen, die Flash Player 10.1 und höher unterstützen. Siehe „Ausführen von ActionScript 3.0-Beispielen auf Mobilgeräten“ auf Seite 1181. Sie können diese Beispiele mit Flash Professional oder Flash Builder auch auf Ihrem Computer ausführen.

TV-Geräte Diese Beispiele lassen sich zwar nicht auf TV-Geräten ausführen, Sie können aber auch Nutzen daraus ziehen, wenn Sie sie auf dem Computer ausführen. Unter Flash Platform for TV auf der Adobe Developer Connection-Website finden Sie Informationen zum Entwickeln von Anwendungen für TV-Geräte.

Beispielarten

Folgende Arten von ActionScript 3.0-Codebeispielen sind verfügbar:

- Codefragmentbeispiele (in der gesamten Dokumentation zu ActionScript 3.0)
- klassenbasierte Beispiele (hauptsächlich im ActionScript Referenzhandbuch für die Adobe Flash-Plattform)
- praktische Beispiele mit mehreren Quelldateien (laden Sie die ZIP-Dateien von www.adobe.com/go/learn_programmingAS3samples_flash_de herunter)

Beispiele mit Codefragmenten

Ein Codefragment-Beispiel sieht folgendermaßen aus:

```
var x:int = 5;
trace(x); // 5
```

Codefragmente enthalten gerade genug Code, um ein einzelnes Konzept zu veranschaulichen. Sie enthalten normalerweise keine Paket- oder Klassenanweisungen.

Klassenbasierte Beispiele

Viele Beispiele zeigen den Quellcode für eine ganze ActionScript-Klasse. Ein klassenbasiertes Beispiel sieht folgendermaßen aus:

```
package {
    public class Example1 {
        public function Example1():void {
            var x:int = 5;
            trace(x); //5
        }
    }
}
```

Der Code eines klassenbasierten Beispiels umfasst eine Paketanweisung, eine Klassendeklaration und eine Konstruktorfunktion.

Praktische Beispiele, die mehrere Quelldateien enthalten

Zahlreiche Themen im ActionScript 3.0 Entwicklerhandbuch werden mit praktischen Beispielen abgeschlossen, die zeigen, wie bestimmte ActionScript-Funktionen in der Praxis verwendet werden. Diese Beispiele umfassen meist mehrere Dateien, wie zum Beispiel:

- Mindestens eine ActionScript-Quelldatei
- Eine FLA-Datei zur Verwendung mit Flash Professional
- Mindestens eine MXML-Datei zur Verwendung mit Flash Builder
- Datendateien, Bilddateien, Sounddateien und andere Elemente, die von der Beispielanwendung genutzt werden (optional)

Praktische Beispiele werden normalerweise in Form von ZIP-Archivdateien bereitgestellt.

Liste der Beispiele für Entwickler in der ZIP-Datei

Die ZIP-Datei für Flash Professional CS5 und Flex 4 (Download von www.adobe.com/go/learn_programmingAS3samples_flash_de) enthält die folgenden Beispiele:

- AlarmClock („Beispiel für die Ereignisverarbeitung: Alarm Clock“ auf Seite 150)
- AlgorithmicVisualGenerator („Beispiel für die Zeichnungs-API: Algorithmic Visual Generator“ auf Seite 247)
- ASCIIArt („Strings-Beispiel: ASCII-Grafik“ auf Seite 20)
- CapabilitiesExplorer („Capabilities-Beispiel: Ermitteln von Systemeigenschaften“ auf Seite 930)
- CustomErrors („Beispiel für die Fehlerverarbeitung: CustomErrors-Anwendung“ auf Seite 74)
- DisplayObjectTransformer („Geometrie-Beispiel: Anwenden einer Matrixtransformation auf ein Anzeigeobjekt“ auf Seite 232)
- FilterWorkbench („Beispiel für das Filtern von Anzeigeobjekten: Filter Workbench“ auf Seite 312)
- GlobalStockTicker („Beispiel: Internationalisierung einer Börsenticker-Anwendung“ auf Seite 1018)
- IntrovertIM_HTML („Beispiel für externe API: Kommunikation zwischen ActionScript und JavaScript in einem Webbrowser“ auf Seite 904)
- NewsLayout („TextField-Beispiel: Textformatierung im Zeitungsstil“ auf Seite 415)
- PlayList („Arrays-Beispiel: PlayList“ auf Seite 50)
- PodcastPlayer („Sound-Beispiel: Podcast-Player“ auf Seite 498)
- ProjectionDragger („Beispiel: Perspektivische Projektion“ auf Seite 382)
- ReorderByZ („Verwenden von 3DMatrix-Objekten zur Neuanordnung der Anzeige“ auf Seite 387)
- RSSViewer („Beispiel für XML in ActionScript: Laden von RSS-Daten aus dem Internet“ auf Seite 121)

- RuntimeAssetsExplorer („Movieclip-Beispiel: RuntimeAssetsExplorer“ auf Seite 353)
- SimpleClock („Beispiel für Datum und Uhrzeit: Einfache Analoguhr“ auf Seite 6)
- SpinningMoon („Bitmap-Beispiel: Animated Spinning Moon“ auf Seite 271
- SpriteArranger („Beispiel für ein Anzeigeobjekt: SpriteArranger“ auf Seite 217)
- TelnetSocket („TCP-Socket-Beispiel: Erstellen eines Telnet-Clients“ auf Seite 852)
- VideoJukebox („Video-Beispiel: Video Jukebox“ auf Seite 539)
- WikiEditor („Beispiel für reguläre Ausdrücke: Wiki-Parser“ auf Seite 97)
- WordSearch („Beispiel für Mauseingabe: WordSearch“ auf Seite 615)

Praktische Beispiele sind auch in vielen Kurzanleitungen im Flash Developer Center und im Flex Developer Center enthalten.

Ausführen von ActionScript 3.0-Beispielen in Flash Professional

Verwenden Sie eines der folgenden Verfahren (je nach Beispieltyp), um ein Beispiel mit Flash Professional auszuführen.

Ausführen eines Codefragment-Beispiels in Flash Professional

So führen Sie ein Codefragment-Beispiel in Flash Professional aus:

1. Wählen Sie „Datei“ > „Neu“.
2. Wählen Sie im Dialogfeld „Neues Dokument“ die Option „Flash-Dokument“ aus und klicken Sie dann auf „OK“.

   Ein neues Flash-Fenster wird angezeigt.
3. Klicken Sie im Bedienfeld „Zeitleiste“ auf das erste Bild auf der ersten Ebene.
4. Geben oder fügen Sie im Bedienfeld „Aktionen“ das Codefragment-Beispiel ein.
5. Wählen Sie „Datei“ > „Speichern“. Benennen Sie die Datei und klicken Sie auf „OK“.
6. Wählen Sie „Steuerung“ > „Film testen“ aus, um das Beispiel zu testen.

Ausführen eines klassenbasierten Beispiels in Flash Professional

So führen Sie ein klassenbasiertes Beispiel in Flash Professional aus:

1. Wählen Sie „Datei“ > „Neu“.
2. Wählen Sie im Dialogfeld „Neues Dokument“ die Option „ActionScript-Datei“ aus und klicken Sie dann auf „OK“. Ein neues Editorfenster wird angezeigt.
3. Kopieren Sie den Code des klassenbasierten Beispiels und fügen Sie ihn in das Editorfenster ein.

   Wenn die Klasse die Hauptdokumentklasse für das Programm ist, muss sie die MovieClip-Klasse erweitern.

```
import flash.display.MovieClip;
public class Example1 extends MovieClip{
//...
}
```

Achten Sie auch darauf, dass alle Klassen, auf die im Beispiel verwiesen wird, mit `import`-Anweisungen deklariert werden.

4 Wählen Sie „Datei“ > „Speichern“ aus. Geben Sie der Datei den Namen der Klasse im Beispiel (zum Beispiel ContextMenuExample.as).

Hinweis: *Einige der klassenbasierten Beispiele, wie das flashx.textLayout.container.ContainerController-Klassenbeispiel, enthalten mehrere Ebenen in der Paketdeklaration (`package flashx.textLayout.container.examples {`). Speichern Sie bei diesen Beispielen die Datei in einem Unterordner, der der Paketdeklaration entspricht (flashx/textLayout/container/examples), oder entfernen Sie den Paketnamen (sodass der ActionScript-Code nur mit `package {` beginnt); Sie können die Datei dann von jedem Ort aus testen.*

5 Wählen Sie „Datei“ > „Neu“.

6 Wählen Sie im Dialogfeld „Neues Dokument“ die Option „Flash-Dokument (ActionScript 3.0)“ aus und klicken Sie dann auf „OK“. Ein neues Flash-Fenster wird angezeigt.

7 Geben Sie im Bedienfeld „Eigenschaften“ im Feld „Dokumentklasse“ den Namen der Beispielklasse ein. Dieser Name sollte mit dem Namen der ActionScript-Quelldatei übereinstimmen, die Sie gerade gespeichert haben (z. B. ContextMenuExample).

8 Wählen Sie „Datei“ > „Speichern“ aus. Geben Sie der FLA-Datei den Namen der Klasse im Beispiel (zum Beispiel ContextMenuExample.fla).

9 Wählen Sie „Steuerung“ > „Film testen“ aus, um das Beispiel zu testen.

Ausführen eines praktischen Beispiels in Flash Professional

Praktische Beispiele werden normalerweise in Form von ZIP-Archivdateien bereitgestellt. So führen Sie ein praktisches Beispiel in Flash Professional aus:

1 Dekomprimieren Sie die Archivdatei in einem Ordner Ihrer Wahl.

2 Wählen Sie in Flash Professional „Datei“ > „Öffnen“ aus.

3 Wechseln Sie zu dem Ordner, in den Sie die Archivdatei dekomprimiert haben. Wählen Sie die FLA-Datei in diesem Ordner aus und klicken Sie auf „Öffnen“.

4 Wählen Sie „Steuerung“ > „Film testen“ aus, um das Beispiel zu testen.

Ausführen von ActionScript 3.0-Beispielen in Flash Builder

Verwenden Sie eines der folgenden Verfahren (je nach Beispieltyp), um ein Beispiel mit Flash Builder auszuführen.

Ausführen eines Codefragment-Beispiels in Flash Builder

So führen Sie ein Codefragment-Beispiel in Flash Builder aus:

1 Erstellen Sie entweder ein neues Flex-Projekt („Datei“ > „Neu“ > „Flex-Projekt“) oder erstellen Sie in einem vorhandenen Flex-Projekt eine neue MXML-Anwendung („Datei“ > „Neu“ > „MXML-Anwendung“). Geben Sie dem Projekt oder der Anwendung einen aussagekräftigen Namen (wie ContextMenuExample).

2 Fügen Sie in der erstellten MXML-Datei ein `<mx:Script>`-Tag hinzu.

3 Fügen Sie den Inhalt des Codefragment-Beispiels zwischen den Tags `<mx:Script>` und `</mx:Script>` ein. Speichern Sie die MXML-Datei.

4 Zum Ausführen des Beispiels wählen Sie „Ausführen“ > „Ausführen“ für die MXML-Hauptdatei (z. B. „Ausführen“ > „Ausführen ContextMenuExample“).

Ausführen eines klassenbasierten Beispiels in Flash Builder

So führen Sie ein klassenbasiertes Beispiel in Flash Builder aus:

1 Wählen Sie „Datei“ > „Neu“ > „ActionScript-Projekt“.

2 Geben Sie den Namen der Primärklasse (wie beispielsweise ContextMenuExample) in das Feld „Projektname“ ein. Verwenden Sie die Standardwerte für die anderen Felder (oder ändern Sie sie je nach Ihrer Umgebung). Klicken Sie auf „Fertig stellen“, um das Projekt und die ActionScript-Hauptdatei zu erstellen.

3 Löschen Sie eventuell generierten Inhalt aus der ActionScript-Datei. Fügen Sie den Beispielcode, einschließlich Paket- und Import-Anweisungen, in die ActionScript-Datei ein und speichern Sie die Datei.

Hinweis: *Einige der klassenbasierten Beispiele, wie das flashx.textLayout.container.ContainerController-Klassenbeispiel, enthalten mehrere Ebenen in der Paketdeklaration (`package flashx.textLayout.container.examples {`). Speichern Sie bei diesen Beispielen die Datei in einem Unterordner, der der Paketdeklaration entspricht (flashx/textLayout/container/examples), oder entfernen Sie den Paketnamen (sodass der ActionScript-Code nur mit `package {` beginnt); Sie können die Datei dann von jedem Ort aus testen.*

4 Zum Ausführen des Beispiels wählen Sie „Ausführen“ > „Ausführen“ für den ActionScript-Hauptklassennamen (z. B. „Ausführen“ > „Ausführen ContextMenuExample“).

Ausführen eines praktischen Beispiels in Flash Builder

Praktische Beispiele werden normalerweise in Form von ZIP-Archivdateien bereitgestellt. So führen Sie ein praktisches Beispiel in Flash Builder aus:

1 Dekomprimieren Sie die Archivdatei in einem Ordner Ihrer Wahl. Geben Sie dem Ordner einen aussagekräftigen Namen (zum Beispiel ContextMenuExample).

2 Wählen Sie in Flash Builder „Datei“ > „Neues Flex-Projekt“ aus. Klicken Sie im Bereich für das Projektverzeichnis auf „Durchsuchen“ und wählen Sie den Ordner aus, der die Beispieldateien enthält. Geben Sie im Feld für den Projektnamen den Ordnernamen ein (z. B. ContextMenuExample). Verwenden Sie die Standardwerte für die anderen Felder (oder ändern Sie sie je nach Ihrer Umgebung). Klicken Sie auf „Weiter“, um fortzufahren.

3 Klicken Sie im Bedienfeld „Ausgabe“ auf „Weiter“, um den Standardwert zu übernehmen.

4 Klicken Sie im Bedienfeld „Quellpfad“ auf die Schaltfläche „Durchsuchen“ neben dem Feld für die Hauptanwendungsdatei. Wählen Sie die MXML-Hauptbeispieldatei im Beispielordner aus. Klicken Sie auf „Fertig stellen“, um die Projektdateien zu erstellen.

5 Zum Ausführen des Beispiels wählen Sie „Ausführen“ > „Ausführen“ für die MXML-Hauptdatei (z. B. „Ausführen“ > „Ausführen ContextMenuExample“).

Ausführen von ActionScript 3.0-Beispielen auf Mobilgeräten

Sie können ActionScript 3.0-Codebeispiele auf Mobilgeräten ausführen, die Flash Player 10.1 unterstützen. Normalerweise führen Sie ein Codebeispiel jedoch aus, um die Funktionsweise bestimmter Klassen und Methoden zu erlernen. In diesem Fall führen Sie das Beispiel nicht auf Mobilgeräten aus, sondern beispielsweise auf einem Desktopcomputer. Auf dem Desktopcomputer können Sie trace-Anweisungen und andere Debugger-Tools in Flash Professional oder Flash Builder verwenden, um ein besseres Verständnis eines Codebeispiels zu erlangen.

Wenn Sie das Beispiel auf einem Mobilgerät ausführen möchten, können Sie die Dateien entweder auf das Gerät oder auf einen Webserver kopieren. Führen Sie die folgenden Schritte aus, um die Dateien auf das Gerät zu kopieren und das Beispiel im Browser auszuführen:

1. Erstellen Sie die SWF-Datei anhand der Anleitungen unter „Ausführen von ActionScript 3.0-Beispielen in Flash Professional“ auf Seite 1178 oder „Ausführen von ActionScript 3.0-Beispielen in Flash Builder“ auf Seite 1179. In Flash Professional erstellen Sie die SWF-Datei, wenn Sie „Steuerung“ > „Film testen“ auswählen. In Flash Builder erstellen Sie die SWF-Datei, wenn Sie das Flash Builder-Projekt ausführen, debuggen oder erstellen.
2. Kopieren Sie die SWF-Datei in ein Verzeichnis auf dem Mobilgerät. Verwenden Sie die Software des Geräts zum Kopieren der Datei.
3. Geben Sie im Browser des Mobilgeräts die file://-URL der SWF-Datei in die Adresszeile ein, wie beispielsweise `file:://applications/myExample.swf`.

Führen Sie folgende Schritte aus, um die Dateien auf einen Webserver zu kopieren und das Beispiel im Browser des Geräts auszuführen:

1. Erstellen Sie eine SWF- und eine HTML-Datei. Folgen Sie zunächst den Anleitungen unter „Ausführen von ActionScript 3.0-Beispielen in Flash Professional“ auf Seite 1178 oder „Ausführen von ActionScript 3.0-Beispielen in Flash Builder“ auf Seite 1179. In Flash Professional wird durch Auswahl von „Steuerung“ > „Film testen“ nur die SWF-Datei erstellt. Um beide Dateien zu erstellen, wählen Sie zunächst im Dialogfeld „Einstellungen für Veröffentlichungen“ auf der Registerkarte „Formate“ sowohl Flash als auch HTML aus. Wählen Sie dann „Datei“ > „Veröffentlichen“ aus, um die HTML-Datei und die SWF-Datei zu erstellen. In Flash Builder erstellen Sie die SWF- und die HTML-Datei, wenn Sie das Flash Builder-Projekt ausführen, debuggen oder erstellen.
2. Kopieren Sie die SWF- und die HTML-Datei in ein Verzeichnis auf dem Webserver.
3. Geben Sie im Browser des Mobilgeräts die HTTP-Adresse der HTML-Datei in die Adresszeile ein, zum Beispiel `http://www.myWebServer/examples/myExample.html`.

Vor dem Ausführen eines Beispiels auf einem Mobilgerät sollten Sie die folgenden Aspekte berücksichtigen.

Bühnengröße

Beim Ausführen eines Beispiels auf einem Mobilgerät wird eine wesentlich kleinere Bühne verwendet als bei einem anderen Gerät. Für viele Beispiele ist keine bestimmte Bühnengröße erforderlich. Geben Sie bei der Erstellung der SWF-Datei eine für das Mobilgerät geeignete Bühnengröße an, wie zum Beispiel 176 x 208 Pixel.

Mit den praktischen Beispielen im ActionScript 3.0 Entwicklerhandbuch sollen verschiedene Konzepte und Klassen von ActionScript 3.0 veranschaulicht werden. Die entsprechenden Benutzeroberflächen wurden so konzipiert, dass sie auf Desktopcomputern und Laptops ansprechend aussehen und gut funktionieren. Obwohl die Beispiele auf Mobilgeräten funktionieren, sind Bühnengröße und Design der Benutzeroberfläche nicht für den kleinen Bildschirm geeignet. Sie sollten die praktischen Beispiele auf einem Computer ausführen, um ActionScript zu erlernen, und dann die zugehörigen Codefragmente in den Mobilanwendungen ausführen.

Textfelder anstelle von trace-Anweisungen

Beim Ausführen eines Beispiels auf einem Mobilgerät können Sie die Ausgabe der trace-Anweisungen des Beispiels nicht sehen. Um die Ausgabe zu sehen, erstellen Sie eine Instanz der TextField-Klasse. Fügen Sie dann den Text der trace-Anweisungen an die `text`-Eigenschaft des Textfeldes an.

Zum Einrichten eines Textfeldes für die trace-Ausgabe können Sie die folgende Funktion verwenden:

```
function createTracingTextField(x:Number, y:Number,
                                width:Number, height:Number):TextField {

    var tracingTF:TextField = new TextField();
    tracingTF.x = x;
    tracingTF.y = y;
    tracingTF.width = width;
    tracingTF.height = height;

    // A border lets you more easily see the area the text field covers.
    tracingTF.border = true;
    // Left justifying means that the right side of the text field is automatically
    // resized if a line of text is wider than the width of the text field.
    // The bottom is also automatically resized if the number of lines of text
    // exceed the length of the text field.
    tracingTF.autoSize = TextFieldAutoSize.LEFT;

    // Use a text size that works well on the device.
    var myFormat:TextFormat = new TextFormat();
    myFormat.size = 18;
    tracingTF.defaultTextFormat = myFormat;

    addChild(tracingTF);
    return tracingTF;
}
```

Fügen Sie diese Funktion beispielsweise der Dokumentklasse als private Funktion hinzu. Dann können Sie in anderen Methoden der Dokumentklasse den folgenden Code verwenden, um Daten zu verfolgen.

```
var traceField:TextField = createTracingTextField(10, 10, 150, 150);
// Use the newline character "\n" to force the text to the next line.
traceField.appendText("data to trace\n");
traceField.appendText("more data to trace\n");
// Use the following line to clear the text field.
traceField.appendText("");
```

Die `appendText()`-Methode akzeptiert nur einen Wert als Parameter. Bei diesem Wert handelt es sich um einen String (eine String-Instanz oder ein String-Literal). Zum Ausgeben des Wertes einer Variablen, die keinen String-Wert enthält, müssen Sie den Wert zuerst in einen String umwandeln. Dies können Sie am einfachsten mit einem Aufruf der `toString()`-Methode des jeweiligen Objekts verwirklichen:

```
var albumYear:int = 1999;
traceField.appendText("albumYear = ");
traceField.appendText(albumYear.toString());
```

Textgröße

Viele Beispiele verwenden Textfelder, damit ein Konzept besser veranschaulicht werden kann. Manchmal lässt sich der Text auf einem Mobilgerät besser lesen, wenn die Größe des Textes im Textfeld angepasst wird. Wenn ein Beispiel eine TextField-Instanz namens `myTextField` enthält, können Sie die Textgröße mit dem folgenden Code ändern:

```
// Use a text size that works well on the device.
var myFormat:TextFormat = new TextFormat();
myFormat.size = 18;
myTextField.defaultTextFormat = myFormat
```

Erfassen von Benutzereingaben

Das Betriebssystem und der Browser des Mobilgeräts erfassen einige Benutzereingabe-Ereignisse, die der SWF-Inhalt nicht erhält. Das konkrete Verhalten richtet sich nach dem Betriebssystem und dem Browser; es kann jedoch ein unerwartetes Verhalten auftreten, wenn die Beispiele auf einem Mobilgerät ausgeführt werden. Weitere Informationen finden Sie unter „KeyboardEvent-Reihenfolge“ auf Seite 598.

Die Benutzeroberflächen zahlreicher Beispiele sind für Desktopcomputer oder Laptops ausgelegt. So eignen sich beispielsweise die meisten praktischen Beispiele im ActionScript 3.0 Entwicklerhandbuch besonders gut für die Anzeige auf einem Desktop. Deshalb kann die Bühne auf dem Bildschirm eines Mobilgeräts nicht immer vollständig angezeigt werden. Ob die Möglichkeit besteht, den Inhalt im Browser zu durchblättern (zu schwenken), richtet sich nach dem Browser. Die Beispiele sind auch nicht dafür ausgelegt, Ereignisse in Bezug auf das Blättern oder den Bildlauf zu erfassen und zu verarbeiten. Deshalb sind die Benutzeroberflächen einiger Beispiele nicht für die Ausführung auf einem kleinen Bildschirm geeignet. Sie sollten die Beispiele auf einem Computer ausführen, um ActionScript zu erlernen, und dann die zugehörigen Codefragmente in den Mobilanwendungen ausführen.

Weitere Informationen finden Sie unter „Schwenken und Scrollen von Anzeigeobjekten“ auf Seite 190.

Steuern des Fokus

Bei manchen Beispielen müssen Sie den Fokus auf ein Feld richten. Dies bietet Ihnen die Möglichkeit, Text einzugeben oder eine Schaltfläche auszuwählen. Um den Fokus auf ein bestimmtes Feld zu richten, verwenden Sie das Zeigegerät des Mobilgeräts oder Ihren Finger. Sie können auch die Navigationstasten des Mobilgeräts verwenden, um den Fokus auf ein Feld zu richten. Um eine Schaltfläche auszuwählen, die derzeit den Fokus hat, verwenden Sie die Auswahltaste des Mobilgeräts, ähnlich wie die Eingabetaste auf einem Computer. Bei manchen Geräten müssen Sie zweimal auf eine Schaltfläche tippen, um sie auszuwählen.

Weitere Informationen zum Fokus finden Sie unter „Fokusverwaltung“ auf Seite 593.

Steuern von Mausereignissen

Zahlreiche Beispiele warten auf Mausereignisse. Auf einem Computer treten Mausereignisse beispielsweise auf, wenn Sie die Maus über ein Anzeigeobjekt bewegen oder auf ein Anzeigeobjekt klicken. Auf Mobilgeräten werden Ereignisse, die über Zeigegeräte oder mit dem Finger ausgelöst werden, als Berührungsereignisse bezeichnet. Flash Player 10.1 ordnet Berührungsereignisse den Mausereignissen zu. Durch diese Zuordnung wird gewährleistet, dass SWF-Inhalt, der vor Flash Player 10.1 entwickelt wurde, auch weiterhin funktioniert. Deswegen funktionieren die Beispiele, wenn Sie ein Anzeigeobjekt mithilfe eines Zeigegeräts auswählen oder ziehen.

Leistung

Mobilgeräte verfügen über weniger Prozessorleistung als Desktopgeräte. Deshalb werden Beispiele, die die CPU stark beanspruchen, auf Mobilgeräten möglicherweise langsamer ausgeführt. So ist für das „Beispiel für die Zeichnungs-API: Algorithmic Visual Generator“ auf Seite 247 bei jedem Bild ein hoher Berechnungs- und Zeichenaufwand erforderlich. Bei Ausführung dieses Beispiels auf einem Computer werden verschiedene Zeichnungs-APIs vorgestellt. Wegen der eingeschränkten Leistung ist das Beispiel jedoch nicht für alle Mobilgeräte geeignet.

Weitere Informationen zur Leistung auf Mobilgeräten finden Sie unter *Leistungsoptimierung für die Flash-Plattform*.

Empfohlene Verfahren

Die Beispiele berücksichtigen keine empfohlenen Verfahren für die Entwicklung von Anwendungen für Mobilgeräte. Der eingeschränkten Speicher- und Leistungskapazität von Mobilgeräten muss besondere Beachtung eingeräumt werden. Gleichermaßen gelten hinsichtlich der Benutzeroberfläche für einen kleinen Bildschirm andere Anforderungen als für die Anzeige auf einem Desktop. Weitere Informationen zur Entwicklung von Anwendungen für Mobilgeräte *finden Sie unter Leistungsoptimierung für die Flash-Plattform*.

Kapitel 66: SQL-Unterstützung in lokalen Datenbanken

Adobe AIR enthält eine SQL-Datenbankengine mit Unterstützung für lokale SQL-Datenbanken und zahlreiche SQL-Standardfunktionen über das Open-Source-Datenbanksystem SQLite. Die Laufzeitumgebung legt nicht fest, wie oder wo Datenbankdaten im Dateisystem gespeichert werden. Jede Datenbank wird vollständig in einer Datei gespeichert. Entwickler können das Verzeichnis im Dateisystem angeben, in dem die Datenbankdatei gespeichert wird. Eine AIR-Anwendung kann auf eine oder mehrere separate Datenbanken (d. h. separate Datenbankdateien) zugreifen. Dieses Dokument enthält eine Beschreibung der SQL-Syntax und der unterstützten Datentypen für lokale SQL-Datenbanken in Adobe AIR. Das vorliegende Dokument ist nicht als umfassende SQL-Referenz gedacht. Es werden vielmehr spezifische Details des von Adobe AIR unterstützten SQL-Dialekts beschrieben. Die Laufzeitumgebung unterstützt die meisten der SQL-Dialekte des SQL-92-Standards. Es sind zahlreiche Referenzen, Websites, Bücher und Schulungsmaterialien zum Erlernen von SQL verfügbar. Das vorliegende Dokument soll nicht als umfassende SQL-Referenz oder als Lernprogramm dienen. Stattdessen beschäftigt sich dieses Dokument hauptsächlich mit der von AIR unterstützten SQL-Syntax und den Unterschieden zwischen SQL 92 und den unterstützten SQL-Dialekten.

Konventionen für SQL-Anweisungsdefinitionen

In den Anweisungsdefinitionen in diesem Dokument gelten die folgenden Konventionen:

- Groß- und Kleinschreibung
  - GROSSBUCHSTABEN – SQL-Literalschlüsselwörter werden ganz in Großbuchstaben geschrieben.
  - KLEINBUCHSTABEN – Platzhalterbegriffe oder Namen von Klauseln werden ganz in Kleinbuchstaben geschrieben.
- Definitionszeichen
  - ::= Kennzeichnet eine Klausel- oder Anweisungsdefinition.
- Gruppierung und alternative Zeichen
  - | Der senkrechte Strich steht zwischen Alternativoptionen und kann als „oder“ interpretiert werden.
  - [] Elemente in eckigen Klammern sind optional; die eckigen Klammern können ein einzelnes Element oder einen Satz an Alternativelementen enthalten.
  - () Runde Klammern um eine Menge an Alternativelementen (eine Menge von Elementen, die durch senkrechte Striche getrennt sind) kennzeichnen eine erforderliche Elementgruppe, also mehrere Elemente, bei denen es sich um die möglichen Werte für ein einzelnes erforderliches Element handelt.
- Quantifizierer
  - + Ein Pluszeichen nach einem Element in Klammern weist darauf hin, dass das voranstehende Element einmal oder mehrmals auftreten kann.
  - * Ein Sternchen nach einem Element in eckigen Klammern weist darauf hin, dass das voranstehende Element (in eckigen Klammern) 0 Mal oder mehrmals auftreten kann.
- Literale Zeichen
  - * Ein Sternchen in einem Spaltennamen oder zwischen den Klammern nach einem Funktionsnamen bezeichnet ein Literalsternchen anstelle des Quantifizierers für „0 Mal oder mehrmals“.
  - . Ein Punkt steht für einen Literalpunkt.

- , Ein Komma steht für ein Literalkomma.
- () Ein rundes Klammernpaar, das eine einzelne Klausel oder ein einzelnes Element umgibt, weist darauf hin, dass die Klammern erforderliche Literalklammernzeichen sind.
- Andere Zeichen, sofern nicht anders angegeben, stellen diese Literalzeichen dar.

Unterstützte SQL-Syntax

Die folgenden SQL-Syntaxbeispiele werden von der SQL-Datenbankengine in Adobe AIR unterstützt. Die Beispiele sind in verschiedene Anweisungs- und Klauseltypen, Ausdrücke, integrierte Funktionen und Operatoren unterteilt. Es werden die folgenden Themen behandelt:

- Allgemeine SQL-Syntax
- Datenbearbeitungsanweisungen (SELECT, INSERT, UPDATE und DELETE)
- Datendefinitionsanweisungen (CREATE-, ALTER- und DROP-Anweisungen für Tabellen, Indizes, Ansichten und Auslöser)
- Spezielle Anweisungen und Klauseln
- Integrierte Funktionen (Aggregations-, Skalar- und Datums-/Uhrzeitformat-Funktionen)
- Operatoren
- Parameter
- Nicht unterstützte SQL-Funktionen
- Zusätzliche SQL-Funktionen

Allgemeine SQL-Syntax

Zusätzlich zu der spezifischen Syntax für verschiedene Anweisungen und Ausdrücke gelten folgende allgemeine Regeln der SQL-Syntax:

Groß-/Kleinschreibung Die Groß- und Kleinschreibung ist bei SQL-Anweisungen, einschließlich der Objektnamen, nicht wichtig. Allerdings werden SQL-Schlüsselwörter in SQL-Anweisungen häufig in Großbuchstaben geschrieben. Auch im vorliegenden Dokument wird diese Konvention beachtet. Während die Groß- und Kleinschreibung in der SQL-Syntax nicht beachtet werden muss, ist dies bei literalen Textwerten in SQL der Fall. Bei Vergleich- und Sortieroperationen kann die Groß-/Kleinschreibung eine Rolle spielen, wenn dies von der für eine Spalte oder einen Vorgang definierten Überprüfungssequenz so angegeben ist. Weitere Informationen finden Sie unter COLLATE.

Leerraum Einzelne Wörter in einer SQL-Anweisung müssen durch ein Leerraumzeichen (ein Leerzeichen, ein Tabulatorzeichen, eine neue Zeile usw.) getrennt sein. Zwischen Wörtern und Symbolen ist die Verwendung von Leerraum jedoch optional. Art und Anzahl der Leerraumzeichen ist in SQL-Anweisungen nicht relevant. Sie können SQL-Anweisungen mit Leerraumzeichen, zum Beispiel Einzügen und Zeilenumbrüchen, formatieren, um die Lesbarkeit zu verbessern. Die Bedeutung der Anweisung wird dadurch nicht geändert.

Datenbearbeitungsanweisungen

Anweisungen zur Bearbeitung von Daten gehören zu den am häufigsten verwendeten SQL-Anweisungen. Mit diesen Anweisungen werden Daten in Datenbanktabellen abgerufen, hinzugefügt, geändert und entfernt. Die folgenden Datenbearbeitungsanweisungen werden unterstützt: SELECT, INSERT, UPDATE und DELETE.

SELECT

Die SELECT-Anweisung wird zum Abfragen der Datenbank verwendet. Das Ergebnis von SELECT sind null oder mehr Datenzeilen, wobei jede Zeile eine feste Anzahl Spalten enthält. Die Anzahl der Spalten im Ergebnis wird vom result-Spaltennamen oder von der Ausdruckliste zwischen den Schlüsselwörtern SELECT und optional FROM angegeben.

```
sql-statement   ::=  SELECT [ALL | DISTINCT] result
                     [FROM table-list]
                     [WHERE expr]
                     [GROUP BY expr-list]
                     [HAVING expr]
                     [compound-op select-statement]*
                     [ORDER BY sort-expr-list]
                     [LIMIT integer [( OFFSET | , ) integer]]
result          ::=  result-column [, result-column]*
result-column   ::=  * | table-name . * | expr [[AS] string]
table-list      ::=  table [ join-op table join-args ]*
table           ::=  table-name [AS alias] |
                     ( select ) [AS alias]
join-op         ::=  , | [NATURAL] [LEFT | RIGHT | FULL] [OUTER | INNER | CROSS] JOIN
join-args       ::=  [ON expr] [USING ( id-list )]
compound-op     ::=  UNION | UNION ALL | INTERSECT | EXCEPT
sort-expr-list  ::=  expr [sort-order] [, expr [sort-order]]*
sort-order      ::=  [COLLATE collation-name] [ASC | DESC]
collation-name  ::=  BINARY | NOCASE
```

Jeder beliebige Ausdruck kann als Ergebnis verwendet werden. Wenn ein Ergebnisausdruck * ist, werden alle Spalten aller Tabellen für diesen Ausdruck eingesetzt. Wenn der Ausdruck der Name einer Tabelle gefolgt von einem .* ist, besteht das Ergebnis aus allen Spalten in dieser Tabelle.

Das Schlüsselwort DISTINCT führt dazu, dass ein Teilsatz von Ergebniszeilen zurückgegeben wird, in dem sich alle Ergebniszeilen unterscheiden. NULL-Werte werden nicht als verschieden voneinander behandelt. Das Standardverhalten ist, dass alle Ergebniszeilen zurückgegeben werden, was durch das Schlüsselwort ALL ausgedrückt werden kann.

Die Abfrage wird für eine oder mehrere Tabellen ausgeführt, die nach dem Schlüsselwort FROM angegeben sind. Wenn mehrere Tabellennamen durch Kommas getrennt sind, verwendet die Abfrage den Kreuzverbund der einzelnen Tabellen. Mit der JOIN-Syntax kann auch angegeben werden, wie Tabellen verbunden sind. Die einzige Art eines äußeren Verbunds, der unterstützt wird, ist LEFT OUTER JOIN. Der ON-Klauselausdruck in join-args muss in einen booleschen Wert aufgelöst werden. Eine Unterabfrage in Klammern kann als Tabelle in der FROM-Klausel verwendet werden. Die gesamte FROM-Klausel kann ausgelassen werden. In diesem Fall ist das Ergebnis eine einzelne Zeile, die aus den Werten der result-Ausdruckliste besteht.

Die WHERE-Klausel wird verwendet, um die Anzahl der Zeilen zu beschränken, die die Abfrage abruft. WHERE-Klauselausdrücke müssen in einen booleschen Wert aufgelöst werden. Die WHERE-Klauselfilterung wird vor jeglicher Gruppierung ausgeführt. Deshalb können WHERE-Klauselausdrücke keine Aggregationsfunktionen enthalten.

Die GROUP BY-Klausel führt dazu, dass eine oder mehrere Zeilen des Ergebnisses in einer einzelnen Ausgabezeile zusammengefasst werden. Eine GROUP BY-Klausel ist besonders hilfreich, wenn das Ergebnis Aggregationsfunktionen enthält. Die Ausdrücke in der GROUP BY-Klausel brauchen keine Ausdrücke zu sein, die in der SELECT-Ausdruckliste aufgeführt sind.

Die HAVING-Klausel ähnelt WHERE insofern, dass sie die von der Anweisung zurückgegebenen Zeilen beschränkt. Die HAVING-Klausel wird jedoch angewendet, nachdem jegliche von einer GROUP BY-Klausel angegebene Gruppierung ausgeführt wurde. Demzufolge kann der HAVING-Ausdruck auf Werte verweisen, die Aggregationsfunktionen enthalten. Ein HAVING-Klauselausdruck muss nicht in der SELECT-Liste aufgeführt sein. Wie ein WHERE-Ausdruck muss ein HAVING-Ausdruck in einen booleschen Wert aufgelöst werden.

Die ORDER BY-Klausel führt dazu, dass die Ausgabezeilen sortiert werden. Das sort-expr-list-Argument für die ORDER BY-Klausel ist eine Liste von Ausdrücken, die als Schlüssel für die Sortierung verwendet werden. Die Ausdrücke müssen für eine einfache SELECT-Anweisung nicht Teil des Ergebnisses sein, aber in einer zusammengesetzten SELECT-Abfrage (eine SELECT-Abfrage, die einen der compound-op-Operatoren verwendet) muss jeder Sortierausdruck genau mit einer der Ergebnisspalten übereinstimmen. Jeder Sortierausdruck kann optional von einer sort-order-Klausel gefolgt werden, die aus dem Schlüsselwort COLLATE und dem Namen einer Überprüfungsfunktion besteht, die zur Sortierung von Text verwendet wird, und/oder dem Schlüsselwort ASC oder DESC, um die Sortierreihenfolge festzulegen (aufsteigend und absteigend). Wenn die sort-order-Klausel ausgelassen wird, wird die standardmäßige aufsteigende Reihenfolge verwendet. Eine Definition der COLLATE-Klausel und der Überprüfungsfunktionen finden Sie unter COLLATE.

Die LIMIT-Klausel gib eine obere Grenze für die Anzahl der im Ergebnis zurückgegebenen Zeilen zurück. Eine negative LIMIT-Klausel gibt an, dass es keine obere Grenze gibt. Der optional verwendete OFFSET hinter LIMIT legt fest, wie viele Zeilen am Anfang des Ergebnissatzes übersprungen werden. In einer zusammengesetzten SELECT-Abfrage kann die LIMIT-Klausel nur nach der letzten SELECT-Anweisung vorkommen und das Limit wird auf die gesamte Abfrage angewendet. Beachten Sie, dass bei Verwendung des OFFSET-Schlüsselwortes in der LIMIT-Klausel die erste Ganzzahl das Limit und die zweite Ganzzahl der Versatz ist. Wenn anstelle des OFFSET-Schlüsselwortes ein Komma verwendet wird, ist die erste Ganzzahl der Versatz und die zweite Ganzzahl das Limit. Dies mag widersprüchlich erscheinen, ist jedoch beabsichtigt, da so die Kompatibilität mit älteren SQL-Datenbanksystemen verbessert wird.

Eine zusammengesetzte SELECT-Abfrage wird aus zwei oder mehr einfachen SELECT-Anweisungen gebildet, die durch einen der Operatoren UNION, UNION ALL, INTERSECT oder EXCEPT verbunden sind. In einer zusammengesetzten SELECT-Abfrage müssen alle darin enthaltenen SELECT-Anweisungen dieselbe Anzahl von Ergebnisspalten angeben. Es kann nur eine einzelne ORDER BY-Klausel nach der letzten SELECT-Anweisung geben (und vor der einzigen LIMIT-Klausel, falls eine angegeben wird). Die Operatoren UNION und UNION ALL verbinden die Ergebnisse der vorstehenden und folgenden SELECT-Anweisungen in einer Tabelle. Der Unterschied liegt darin, dass mit UNION alle Ergebniszeilen eindeutig sind, mit UNION ALL können sie jedoch doppelt auftreten. Der INTERSECT-Operator nimmt den Schnittpunkt der Ergebnisse der vorstehenden und der folgenden SELECT-Anweisungen. EXCEPT nimmt das Ergebnis der vorstehenden SELECT-Anweisung, nachdem die Ergebnisse der folgenden SELECT-Anweisung entfernt wurden. Wenn drei oder mehr SELECT-Anweisungen zu einer zusammengesetzten Anweisung verbunden werden, werden sie von der ersten zur letzten gruppiert.

Eine Definition erlaubter Ausdrücke finden Sie unter „Ausdrücke“.

Ab AIR 2.5 wird der SQL-CAST-Operator beim Lesen unterstützt, um BLOB-Daten in ByteArray-Objekte in ActionScript zu konvertieren. Der folgende Code liest beispielsweise unformatierte Daten, die nicht im AMF-Format gespeichert sind, und speichert sie in einem ByteArray-Objekt:

```
stmt.text = "SELECT CAST(data AS ByteArray) AS data FROM pictures;";
stmt.execute();
var result:SQLResult = stmt.getResult();
var bytes:ByteArray = result.data[0].data;
```

INSERT

Die INSERT-Anweisung gibt es in zwei grundlegenden Formen. Mit dieser Anweisung werden Tabellen mit Daten gefüllt.

```
sql-statement  ::=  INSERT [OR conflict-algorithm] INTO [database-name.] table-name [(column-
list)] VALUES (value-list) |
                    INSERT [OR conflict-algorithm] INTO [database-name.] table-name [(column-
list)] select-statement
               REPLACE INTO [database-name.] table-name [(column-list)] VALUES (value-list) |
                    REPLACE INTO [database-name.] table-name [(column-list)] select-statement
```

Die erste Form (mit dem VALUES-Schlüsselwort) erstellt eine einzelne neue Zeile in einer vorhandenen Tabelle. Wenn keine column-list angegeben wird, muss die Anzahl der Werte mit der Anzahl der Spalten in der Tabelle übereinstimmen. Wenn eine column-list angegeben wird, muss die Anzahl der Werte mit der Anzahl der angegebenen Spalten übereinstimmen. Tabellenspalten, die nicht in der Spaltenliste aufgeführt sind, werden mit dem Standardwert gefüllt, der beim Erstellen der Tabelle definiert wurde, oder mit NULL, wenn kein Standardwert definiert wurde.

Die zweite Form der INSERT-Anweisung nimmt die Daten aus einer SELECT-Anweisung. Die Anzahl der Spalten im Ergebnis der SELECT-Abfrage muss genau mit der Anzahl der Spalten in der Tabelle übereinstimmen, wenn column-list nicht angegeben wurde, oder sie muss mit der Anzahl der in column-list genannten Spalten übereinstimmen. Für jede neue Zeile im SELECT-Ergebnis erfolgt ein neuer Eintrag in der Tabelle. Die SELECT-Anweisung kann einfach oder zusammengesetzt sein. Eine Definition zulässiger SELECT-Anweisungen finden Sie unter SELECT.

Der optionale conflict-algorithm ermöglicht die Angabe eines alternativen Algorithmus zur Auflösung von Beschränkungskonflikten, der während dieses einen Befehls verwendet wird. Eine Erläuterung und eine Definition der Konfliktalgorithmen finden Sie unter „Spezielle Anweisungen und Klauseln“ auf Seite 1197.

Die beiden REPLACE INTO-Formen der Anweisung entsprechen der Verwendung der standardmäßigen INSERT [OR conflict-algorithm]-Form mit dem REPLACE-Konfliktalgorithmus (d. h. INSERT OR REPLACE...-Form).

Die beiden REPLACE INTO-Formen der Anweisung entsprechen der Verwendung der standardmäßigen INSERT [OR conflict-algorithm]-Form mit dem REPLACE-Konfliktalgorithmus (d. h. INSERT OR REPLACE...-Form).

UPDATE

Der update-Befehl ändert vorhandene Datensätze in einer Tabelle.

```
sql-statement  ::=  UPDATE [database-name.] table-name SET column1=value1, column2=value2,...
[WHERE expr]
```

Der Befehl besteht aus dem UPDATE-Schlüsselwort gefolgt von dem Namen der Tabelle, in der Sie Datensätze aktualisieren möchten. Geben Sie nach dem SET-Schlüsselwort den Namen der Spalte und den Wert, in den die Spalte geändert werden soll, in einer Liste mit Kommas als Trennzeichen an. Der WHERE-Klauselausdruck gibt die Zeilen an, in denen die Datensätze aktualisiert werden.

DELETE

Der delete-Befehl wird verwendet, um Datensätze aus einer Tabelle zu entfernen.

```
sql-statement  ::=  DELETE FROM [database-name.] table-name [WHERE expr]
```

Der Befehl besteht aus dem DELETE FROM-Schlüsselwort gefolgt vom Namen der Tabelle, aus der Datensätze entfernt werden sollen.

Ohne eine WHERE-Klausel werden alle Zeilen der Tabelle entfernt. Wenn eine WHERE-Klausel angegeben wird, werden nur die Zeilen entfernt, auf die der Ausdruck zutrifft. Der WHERE-Klauselausdruck muss in einen booleschen Wert aufgelöst werden. Eine Definition erlaubter Ausdrücke finden Sie unter „Ausdrücke“.

Datendefinitionsanweisungen

Anweisungen zur Datendefinition werden verwendet, um Datenbankobjekte wie Tabellen, Ansichten, Indizes und Auslöser zu erstellen, zu ändern und zu entfernen. Die folgenden Anweisungen zur Datendefinition werden unterstützt:

- Tabellen:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
- Indizes:
  - CREATE INDEX
  - DROP INDEX
- Ansichten:
  - CREATE VIEWS
  - DROP VIEWS
- Auslöser:
  - CREATE TRIGGERS
  - DROP TRIGGERS

CREATE TABLE

Eine CREATE TABLE-Anweisung besteht aus dem CREATE TABLE-Schlüsselwort gefolgt vom Namen der neuen Tabelle und einer Liste von Spaltendefinitionen und -beschränkungen (in Klammern). Der Tabellenname kann ein Bezeichner oder ein String sein.

```
sql-statement      ::=  CREATE [TEMP | TEMPORARY] TABLE [IF NOT EXISTS] [database-name.] table-
name
                        ( column-def [, column-def]* [, constraint]* )
sql-statement      ::=  CREATE [TEMP | TEMPORARY] TABLE [database-name.] table-name AS select-
statement
column-def          ::=  name [type] [[CONSTRAINT name] column-constraint]*
type                ::=  typename | typename ( number ) | typename ( number , number )
column-constraint   ::=  NOT NULL [ conflict-clause ] |
                         PRIMARY KEY [sort-order] [ conflict-clause ] [AUTOINCREMENT] |
                         UNIQUE [conflict-clause] |
                         CHECK ( expr ) |
                         DEFAULT default-value |
                         COLLATE collation-name
constraint          ::=  PRIMARY KEY ( column-list ) [conflict-clause] |
                         UNIQUE ( column-list ) [conflict-clause] |
                         CHECK ( expr )
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-algorithm  ::=  ROLLBACK | ABORT | FAIL | IGNORE | REPLACE
default-value      ::=  NULL | string | number | CURRENT_TIME | CURRENT_DATE | CURRENT_TIMESTAMP
sort-order          ::=  ASC | DESC
collation-name      ::=  BINARY | NOCASE
column-list         ::=  column-name [, column-name]*
```

Jede Spaltendefinition ist der Name der Spalte gefolgt vom Datentyp für diese Spalte, danach ein oder mehrere optionale Spaltenbeschränkungen. Der Datentyp für die Spalte bestimmt, welche Daten in dieser Spalte gespeichert werden können. Wird versucht, einen Wert in einer Spalte mit einem anderen Datentyp zu speichern, konvertiert die Laufzeitumgebung den Wert in den geeigneten Typ, falls möglich, oder gibt einen Fehler aus. Weitere Informationen finden Sie unter „Unterstützte Datentypen".

Die NOT NULL-Spaltenbeschränkung gibt an, dass die Spalte keine NULL-Werte enthalten darf.

Eine UNIQUE-Beschränkung führt dazu, dass eine Indexposition für die angegebene(n) Spalte(n) erstellt wird. Dieser Index muss eindeutige Schlüssel enthalten, das heißt, zwei Zeilen dürfen nicht doppelte Werte oder Wertkombinationen für die angegebene(n) Spalte(n) enthalten. Eine CREATE TABLE-Anweisung kann über mehrere UNIQUE-Beschränkungen verfügen, darunter mehrere Spalten mit einer UNIQUE-Beschränkung in der Spaltendefinition und/oder mehrere UNIQUE-Beschränkungen auf Tabellenebene.

Eine CHECK-Beschränkung definiert einen Ausdruck, der evaluiert wird und „true" sein muss, damit die Daten einer Zeile eingefügt oder aktualisiert werden. Der CHECK-Ausdruck muss in einen booleschen Wert aufgelöst werden.

Eine COLLATE-Klausel in einer Spaltendefinition gibt an, welche Textüberprüfungsfunktion verwendet wird, wenn Texteinträge für die Spalte verglichen werden. Standardmäßig wird die BINARY-Überprüfungsfunktion verwendet. Ausführliche Informationen zur COLLATE-Klausel und zu Überprüfungsfunktionen finden Sie unter COLLATE.

Die DEFAULT-Beschränkung definiert einen Standardwert, der beim Ausführen von INSERT verwendet wird. Der Wert kann NULL, eine Stringkonstante oder eine Zahl sein. Der Standardwert kann auch eines der speziellen Schlüsselwörter sein, bei denen die Groß- und Kleinschreibung keine Rolle spielt: CURRENT_TIME, CURRENT_DATE oder CURRENT_TIMESTAMP. Wenn der Wert NULL, eine Stringkonstante oder eine Zahl ist, wird er in die Spalte eingefügt, wenn eine INSERT-Anweisung keinen Wert für die Spalte angibt. Wenn der Wert CURRENT_TIME, CURRENT_DATE oder CURRENT_TIMESTAMP ist, wird das aktuelle Datum und/oder die aktuelle Uhrzeit im UTC-Format in die Spalte eingefügt. Für CURRENT_TIME lautet das Format HH:MM:SS. Für CURRENT_DATE lautet das Format YYYY-MM-DD. Das Format für CURRENT_TIMESTAMP ist YYYY-MM-DD HH:MM:SS.

Durch die Angabe eines PRIMARY KEY wird normalerweise lediglich eine UNIQUE-Indexposition für die entsprechende(n) Spalte(n) erstellt. Wenn die PRIMARY KEY-Beschränkung jedoch für eine einzelne Spalte gilt, die den INTEGER-Datentyp (oder eines seiner Synonyme, wie z. B. int) aufweist, wird diese Spalte intern als eigentlicher Primärschlüssel für die Tabelle verwendet. Dies bedeutet, dass die Spalte nur eindeutige Ganzzahlwerte enthalten kann. (Beachten Sie, dass bei zahlreichen SQLite-Implementierungen nur der INTEGER-Spaltentyp dafür verantwortlich ist, dass die Spalte der interne Primärschlüssel ist, dass in Adobe AIR dieses Verhalten jedoch auch durch Synonyme von INTEGER, wie z. B. int, festgelegt wird.)

Wenn eine Tabelle keine INTEGER PRIMARY KEY-Spalte hat, wird automatisch ein Ganzzahlschlüssel generiert, wenn eine Zeile eingefügt wird. Auf den Primärschlüssel für eine Zeile kann immer mit einem der speziellen Namen ROWID, OID oder _ROWID_ zugegriffen werden. Diese Namen können verwendet werden unabhängig davon, ob es sich um einen ausdrücklich deklarierten INTEGER PRIMARY KEY oder einen intern generierten Wert handelt. Wenn die Tabelle jedoch über einen ausdrücklichen INTEGER PRIMARY KEY verfügt, ist der Name der Spalte in den Ergebnisdaten der tatsächliche Spaltenname, nicht der spezielle Name.

Eine INTEGER PRIMARY KEY-Spalte kann auch das AUTOINCREMENT-Schlüsselwort enthalten. Wenn das AUTOINCREMENT-Schlüsselwort verwendet wird, generiert die Datenbank automatisch einen fortlaufend inkrementierten Ganzzahlschlüssel und fügt ihn in die INTEGER PRIMARY KEY-Spalte ein, wenn eine INSERT-Anweisung keinen ausdrücklichen Wert für die Spalte festlegt.

Es kann nur eine PRIMARY KEY-Beschränkung in einer CREATE TABLE-Anweisung geben. Es kann sich um einen Teil einer Spaltendefinition handeln oder um eine PRIMARY KEY-Beschränkung auf Tabellenebene. Ein Primärschlüssel ist implizit NOT NULL.

Die optionale conflict-clause, die vielen Beschränkungen nachgestellt ist, ermöglicht die Angabe eines alternativen Algorithmus für die Auflösung von Beschränkungskonflikten für diese Beschränkung. Der Standardwert lautet ABORT. Unterschiedliche Beschränkungen innerhalb derselben Tabelle können unterschiedliche Algorithmen für die Konfliktauflösung aufweisen. Wenn eine INSERT- oder UPDATE-Anweisung einen anderen Algorithmus für die Konfliktauflösung angibt, wird dieser Algorithmus anstelle des in der CREATE TABLE-Anweisung angegebenen Algorithmus verwendet. Weitere Informationen finden Sie im Abschnitt zu ON CONFLICT unter „Spezielle Anweisungen und Klauseln“ auf Seite 1197.

Zusätzliche Beschränkungen, zum Beispiel FOREIGN KEY-Beschränkungen, führen zwar nicht zu einem Fehler, werden in der Laufzeitumgebung jedoch ignoriert.

Wenn das TEMP- oder TEMPORARY-Schlüsselwort zwischen CREATE und TABLE vorkommt, ist die erstellte Tabelle nur innerhalb derselben Datenbankverbindung (SQLConnection-Instanz) sichtbar. Sie wird automatisch gelöscht, wenn die Datenbankverbindung beendet wird. Alle Indizes, die für eine temporäre Tabelle erstellt werden, sind ebenfalls temporär. Temporäre Tabellen und Indizes werden in einer separaten Datei gespeichert, die sich von der Hauptdatenbankdatei unterscheidet.

Wenn das optionale database-name-Präfix angegeben wird, wird die Tabelle in einer genannten Datenbank erstellt. (Eine Datenbank, die mit der SQLConnection-Instanz verbunden wurde, indem die attach()-Methode mit dem angegebenen Datenbanknamen aufgerufen wurde.) Es ist ein Fehler, sowohl das database-name-Präfix als auch das TEMP-Schlüsselwort anzugeben, es sei denn, das database-name-Präfix lautet „temp“. Wenn keine Datenbank angegeben wird und das TEMP-Schlüsselwort nicht vorhanden ist, wird die Tabelle in der Hauptdatenbank erstellt. (Dies ist die Datenbank, die über die open()- oder openAsync()-Methode mit der SQLConnection-Instanz verbunden wurde.)

Es gibt keine willkürlichen Grenzen für die Anzahl der Spalten oder die Anzahl der Beschränkungen in einer Tabelle. Auch die Datenmenge in einer Zeile ist nicht willkürlich begrenzt.

Die CREATE TABLE AS-Form definiert die Tabelle als Ergebnissatz einer Abfrage. Die Namen der Tabellenspalten sind die Namen der Spalten im Ergebnis.

Wenn die optionale IF NOT EXISTS-Klausel vorhanden ist und eine andere Tabelle mit demselben Namen bereits existiert, ignoriert die Datenbank den CREATE TABLE-Befehl.

Eine Tabelle kann mit der DROP TABLE-Anweisung entfernt werden und begrenzte Änderungen sind mit der ALTER TABLE-Anweisung möglich.

ALTER TABLE

Der ALTER TABLE-Befehl ermöglicht es dem Benutzer, eine vorhandene Tabelle umzubenennen oder ihr eine neue Spalte hinzuzufügen. Es ist nicht möglich, eine Spalte aus einer Tabelle zu entfernen.

```
sql-statement ::= ALTER TABLE [database-name.] table-name alteration
alteration    ::= RENAME TO new-table-name
alteration    ::= ADD [COLUMN] column-def
```

Die RENAME TO-Syntax wird verwendet, um die durch [database-name.] table-name angegebene Tabelle in new-table-name umzubenennen. Mit diesem Befehl kann eine Tabelle nicht zwischen verbundenen Datenbanken verschoben werden. Es ist nur möglich, eine Tabelle innerhalb derselben Datenbank umzubenennen.

Wenn die umbenannte Tabelle Auslöser oder Indizes aufweist, bleiben diese auch nach der Umbenennung mit der Tabelle verbunden. Gibt es jedoch Ansichtsdefinitionen oder Anweisungen, die von Auslösern ausgeführt werden, die sich auf die umbenannte Tabelle beziehen, werden diese nicht automatisch so geändert, dass sie den neuen Tabellennamen verwenden. Wenn mit einer umbenannten Tabelle Ansichten oder Auslöser verknüpft sind, müssen Sie die Auslöser oder Ansichtsdefinitionen entfernen und unter Verwendung des neuen Tabellennamens neu erstellen.

Mit der ADD [COLUMN]-Syntax wird einer vorhandenen Tabelle eine neue Spalte hinzugefügt. Die neue Spalte wird immer an das Ende der Liste vorhandener Spalten angehängt. Die column-def-Klausel kann jede beliebige Form annehmen, die in einer CREATE TABLE-Anweisung zulässig ist. Dabei gelten die folgenden Einschränkungen:

- Die Spalte darf keine PRIMARY KEY- oder UNIQUE-Beschränkung aufweisen.
- Die Spalte darf keinen Standardwert von CURRENT_TIME, CURRENT_DATE oder CURRENT_TIMESTAMP aufweisen.
- Wenn eine NOT NULL-Beschränkung angegeben ist, muss die Spalte einen anderen Standardwert als NULL haben.

Die Ausführungszeit der ALTER TABLE-Anweisung wird von der Datenmenge in der Tabelle nicht beeinflusst.

DROP TABLE

Die DROP TABLE-Anweisung entfernt eine Tabelle, die mit einer CREATE TABLE-Anweisung erstellt wurde. Die Tabelle mit dem angegebenen table-name ist die Tabelle, die entfernt wird. Sie wird vollständig aus der Datenbank und aus der Festplattendatei entfernt. Die Tabelle kann nicht wiederhergestellt werden. Alle der Tabelle zugewiesenen Indizes werden ebenfalls gelöscht.

```
sql-statement  ::=  DROP TABLE [IF EXISTS] [database-name.] table-name
```

Standardmäßig verringert die DROP TABLE-Anweisung nicht die Größe der Datenbankdatei. Leerer Speicherplatz in der Datenbank wird beibehalten und in nachfolgenden INSERT-Vorgängen verwendet. Mit der SQLConnection.clean()-Methode können Sie Speicherplatz in der Datenbank freigeben. Wenn der autoClean-Parameter beim Erstellen der Datenbank mit dem Wert „true“ belegt wurde, wird der Speicherplatz automatisch freigegeben.

Die optionale IF EXISTS-Klausel unterdrückt den Fehler, der normalerweise ausgegeben wird, wenn die Tabelle nicht vorhanden ist.

CREATE INDEX

Der CREATE INDEX-Befehl besteht aus den CREATE INDEX-Schlüsselwörtern gefolgt von dem Namen des neuen Index, dem ON-Schlüsselwort, dem Namen einer zuvor erstellten Tabelle, die indiziert werden soll, und einer Liste in Klammern, in der die Namen der Tabellenspalten aufgeführt sind, die für den Indexschlüssel verwendet werden sollen.

```
sql-statement  ::=  CREATE [UNIQUE] INDEX [IF NOT EXISTS] [database-name.] index-name
                    ON table-name ( column-name [, column-name]* )
column-name    ::=  name [COLLATE collation-name] [ASC | DESC]
```

Jedem Spaltennamen kann das ASC- oder DESC-Schlüsselwort folgen, um die Sortierreihenfolge anzugeben, die von der Laufzeitumgebung jedoch ignoriert wird. Die Sortierung erfolgt immer in aufsteigender Reihenfolge.

Die COLLATE-Klausel, die jedem Spaltennamen nachgestellt ist, definiert eine Überprüfungssequenz, die für Textwerte in den Spalten verwendet wird. Die Standardüberprüfungssequenz ist die, die für diese Spalte in der CREATE TABLE-Anweisung definiert ist. Wenn keine Überprüfungssequenz angegeben wurde, wird BINARY verwendet. Eine Definition der COLLATE-Klausel und der Überprüfungsfunktionen finden Sie unter COLLATE.

Es gibt keine willkürlichen Grenzen für die Anzahl der Indizes, die einer einzelnen Tabelle zugeordnet werden können. Auch für die Anzahl der Spalten in einer Indexposition gibt es keine Grenzen.

DROP INDEX

Die drop index-Anweisung entfernt einen Index, der mit der CREATE INDEX-Anweisung hinzugefügt wurde. Die angegebene Indexposition wird vollständig aus der Datenbankdatei entfernt. Die Indexposition kann nur wiederhergestellt werden, indem der entsprechende CREATE INDEX-Befehl erneut eingegeben wird.

```
sql-statement ::= DROP INDEX [IF EXISTS] [database-name.] index-name
```

Standardmäßig verringert die DROP INDEX-Anweisung nicht die Größe der Datenbankdatei. Leerer Speicherplatz in der Datenbank wird beibehalten und in nachfolgenden INSERT-Vorgängen verwendet. Mit der SQLConnection.clean()-Methode können Sie Speicherplatz in der Datenbank freigeben. Wenn der autoClean-Parameter beim Erstellen der Datenbank mit dem Wert „true“ belegt wurde, wird der Speicherplatz automatisch freigegeben.

CREATE VIEW

Der CREATE VIEW-Befehl weist einer vordefinierten SELECT-Anweisung einen Namen zu. Dieser neue Name kann dann in einer FROM-Klausel einer anderen SELECT-Anweisung anstelle des Tabellennamens verwendet werden. Ansichten werden häufig zur Vereinfachung von Abfragen verwendet, indem ein komplexer (und häufig verwendeter) Satz von Daten in einer Struktur kombiniert wird, die in anderen Operationen verwendet werden kann.

```
sql-statement ::= CREATE [TEMP | TEMPORARY] VIEW [IF NOT EXISTS] [database-name.] view-name AS
select-statement
```

Wenn das TEMP- oder TEMPORARY-Schlüsselwort zwischen CREATE und VIEW auftritt, ist die erstellte Ansicht nur für die SQLConnection-Instanz sichtbar, die die Datenbank geöffnet hat, und sie wird automatisch gelöscht, wenn die Datenbank geschlossen wird.

Wenn ein [database-name] angegeben wird, wird die Ansicht in der genannten Datenbank erstellt (eine Datenbank, die unter Verwendung der attach()-Methode mit dem angegebenen name-Argument mit der SQLConnection-Instanz verbunden wurde. Es ist ein Fehler, sowohl [database-name] als auch das TEMP-Schlüsselwort anzugeben, es sei denn, [database-name] lautet „temp“. Wenn keine Datenbank angegeben wird und das TEMP-Schlüsselwort nicht vorhanden ist, wird die Ansicht in der Hauptdatenbank erstellt. (Dies ist die Datenbank, die über die open()- oder openAsync()-Methode mit der SQLConnection-Instanz verbunden wurde.)

Ansichten sind schreibgeschützt. Eine DELETE-, INSERT- oder UPDATE-Anweisung kann nicht für eine Ansicht verwendet werden, es sei denn, es wurde mindestens ein Auslöser des zugeordneten Typs (INSTEAD OF DELETE, INSTEAD OF INSERT, INSTEAD OF UPDATE) definiert. Informationen zum Erstellen eines Auslösers für eine Ansicht finden Sie unter CREATE TRIGGER.

Eine Ansicht kann mit der DROP VIEW-Anweisung aus einer Datenbank entfernt werden.

DROP VIEW

Die DROP VIEW-Anweisung entfernt eine Ansicht, die mit einer CREATE VIEW-Anweisung erstellt wurde.

```
sql-statement ::= DROP VIEW [IF EXISTS] view-name
```

Der angegebene view-name ist der Name der zu entfernenden Ansicht. Sie wird aus der Datenbank entfernt, die Daten in den zugrundeliegenden Tabellen werden jedoch nicht geändert.

CREATE TRIGGER

Die create trigger-Anweisung wird verwendet, um dem Datenbankschema Auslöser hinzuzufügen. Ein Auslöser ist eine Datenbankoperation (trigger-action), die automatisch ausgeführt wird, wenn ein angegebenes Datenbankereignis (database-event) auftritt.

```
sql-statement   ::=  CREATE [TEMP | TEMPORARY] TRIGGER [IF NOT EXISTS] [database-name.] trigger-
name
                      [BEFORE | AFTER] database-event
                      ON table-name
                      trigger-action
sql-statement   ::=  CREATE [TEMP | TEMPORARY] TRIGGER [IF NOT EXISTS] [database-name.] trigger-
name
                      INSTEAD OF database-event
                      ON view-name
                      trigger-action
database-event  ::=  DELETE |
                      INSERT |
                      UPDATE |
                      UPDATE OF column-list
trigger-action  ::=  [FOR EACH ROW] [WHEN expr]
                      BEGIN
                        trigger-step ;
                        [ trigger-step ; ]*
                      END
trigger-step    ::=  update-statement |
                      insert-statement |
                      delete-statement |
                      select-statement
column-list     ::=  column-name [, column-name]*
```

Ein Auslöser wird ausgelöst, wenn eine DELETE-, INSERT- oder UPDATE-Anweisung einer bestimmten Datenbanktabelle auftritt oder wenn eine UPDATE-Anweisung von einer oder mehreren angegebenen Spalten einer Tabelle aktualisiert wird. Auslöser sind dauerhaft, sofern nicht das TEMP- oder TEMPORARY-Schlüsselwort verwendet wird. In diesem Fall wird der Auslöser entfernt, wenn die Hauptdatenbankverbindung der SQLConnection-Instanz geschlossen wird. Wenn kein Timing angegeben wird (BEFORE oder AFTER), wird standardmäßig BEFORE für den Auslöser verwendet.

Es werden nur FOR EACH ROW-Auslöser unterstützt, der Text FOR EACH ROW ist deshalb optional. Mit einem FOR EACH ROW-Auslöser werden die trigger-step-Anweisungen für jede Datenbankzeile ausgeführt, die von der Anweisung, die den Auslöser auslöst, eingefügt, aktualisiert oder gelöscht wird, wenn der WHEN-Klauselausdruck zu „true“ evaluiert wird.

Wenn eine WHEN-Klausel angegeben wird, werden die als Auslöserschritte angegebenen SQL-Anweisungen nur für Zeilen ausgeführt, für die die WHEN-Klausel mit „true“ belegt ist. Wenn keine WHEN-Klausel angegeben ist, werden die SQL-Anweisungen für alle Zeilen ausgeführt.

Im Hauptteil eines Auslösers (trigger-action-Klausel) sind die Werte der betroffenen Tabelle vor und nach der Änderung verfügbar, wenn die speziellen Tabellennamen OLD und NEW verwendet werden. Die Struktur der OLD- und NEW-Tabellen stimmt mit der Struktur der Tabelle überein, für die der Auslöser erstellt wurde. Die OLD-Tabelle enthält alle Zeilen, die von der auslösenden Anweisung geändert oder gelöscht wurden, in ihrem Zustand vor den Operationen der auslösenden Anweisung. Die NEW-Tabelle enthält alle Zeilen, die von der auslösenden Anweisung geändert oder erstellt wurden, in ihrem Zustand nach den Operationen der auslösenden Anweisung. Sowohl die WHEN-Klausel als auch die trigger-step-Anweisungen können auf Werte aus den eingefügten, gelöschten oder aktualisierten Zeilen zugreifen, indem Verweise in der Form NEW.column-name und OLD.column-name verwendet werden, wobei column-name der Name einer Spalte in der Tabelle ist, mit der der Auslöser verknüpft ist. Die Verfügbarkeit der OLD- und NEW-Tabellenverweise richtet sich nach dem database-event-Typ, den der Auslöser verarbeitet:

- INSERT – NEW-Verweise sind gültig
- UPDATE – NEW- und OLD-Verweise sind gültig

- DELETE – OLD-Verweise sind gültig

Das angegebene Timing (BEFORE, AFTER oder INSTEAD OF) bestimmt, wann die trigger-step-Anweisungen in Relation zum Einfügen, Ändern oder Entfernen der zugeordneten Zeile ausgeführt werden. Eine ON CONFLICT-Klausel kann als Teil einer UPDATE- oder INSERT-Anweisung in einem trigger-step angegeben werden. Wenn jedoch eine ON CONFLICT-Klausel als Teil der Anweisung angegeben wird, die den Auslöser auslöst, wird stattdessen diese Konfliktverwaltungsrichtlinie verwendet.

Zusätzlich zu Tabellenauslösern kann in einer Ansicht ein INSTEAD OF-Auslöser erstellt werden. Wenn in einer Ansicht ein oder mehrere INSTEAD OF INSERT-, INSTEAD OF DELETE- oder INSTEAD OF UPDATE-Auslöser definiert sind, wird es nicht als Fehler betrachtet, den zugeordneten Anweisungstyp (INSERT, DELETE oder UPDATE) für die Ansicht auszuführen. In diesem Fall wird der zugeordnete Auslöser durch die Ausführung von INSERT, DELETE oder UPDATE für die Ansicht ausgelöst. Da es sich bei dem Auslöser um einen INSTEAD OF-Auslöser handelt, werden die der Ansicht zugrunde liegenden Tabellen nicht von der Anweisung geändert, die den Auslöser auslöst. Die Auslöser können jedoch verwendet werden, um Änderungen an den zugrunde liegenden Tabellen vorzunehmen.

Beim Erstellen eines Auslösers für eine Tabelle mit einer INTEGER PRIMARY KEY-Spalte ist ein wichtiger Punkt zu beachten. Wenn ein BEFORE-Auslöser die INTEGER PRIMARY KEY-Spalte einer Zeile ändert, die von der Anweisung aktualisiert werden soll, die den Auslöser auslöst, findet diese Aktualisierung nicht statt. Dies lässt sich umgehen, indem die Tabelle mit einer PRIMARY KEY-Spalte anstatt mit einer INTEGER PRIMARY KEY-Spalte erstellt wird.

Ein Auslöser kann mit der DROP TRIGGER-Anweisung entfernt werden. Wenn eine Tabelle oder Ansicht entfernt wird, werden automatisch auch alle Auslöser entfernt, die dieser Tabelle oder Ansicht zugeordnet sind.

RAISE()-Funktion

Die spezielle SQL-Funktion RAISE() kann in einer trigger-step-Anweisung eines Auslösers verwendet werden. Diese Funktion hat die folgende Syntax:

```
raise-function  ::=  RAISE ( ABORT, error-message ) |
                     RAISE ( FAIL, error-message ) |
                     RAISE ( ROLLBACK, error-message ) |
                     RAISE ( IGNORE )
```

Wenn während der Auslöserausführung eine der drei ersten Formen aufgerufen wird, wird die angegebene ON CONFLICT-Verarbeitungsaktion (ABORT, FAIL oder ROLLBACK) ausgeführt und die Ausführung der aktuellen Anweisung wird beendet. ROLLBACK wird als fehlgeschlagene Ausführung der Anweisung betrachtet. Deshalb löst die SQLStatement-Instanz, deren execute()-Methode ausgeführt wurde, ein error-Ereignis (SQLErrorEvent.ERROR) aus. Das SQLError-Objekt in der error-Eigenschaft des ausgelösten Ereignisobjekts hat für die details-Eigenschaft den Wert, der in der error-message eingestellt ist, die in der RAISE()-Funktion festgelegt ist.

Wenn RAISE(IGNORE) aufgerufen wird, werden der Rest des aktuellen Auslösers, die Anweisung, die die Ausführung des Auslösers ausgelöst hat, und alle nachfolgenden Auslöser, die ausgeführt würden, verworfen. Es werden keine Datenbankänderungen zurückgenommen. Wenn die Anweisung, die die Ausführung des Auslösers verursacht hat, selbst Teil eines Auslösers ist, nimmt dieses Auslöserprogramm seine Ausführung zu Beginn des nächsten Schritts wieder auf. Weitere Informationen zu den Algorithmen für die Konfliktauflösung finden Sie im Abschnitt ON CONFLICT (Konfliktalgorithmen).

DROP TRIGGER

Die DROP TRIGGER-Anweisung entfernt einen Auslöser, der von der CREATE TRIGGER-Anweisung erstellt wurde.

```
sql-statement  ::=  DROP TRIGGER [IF EXISTS] [database-name.] trigger-name
```

Der Auslöser wird aus der Datenbank entfernt. Beachten Sie, dass Auslöser automatisch entfernt werden, wenn die zugeordnete Tabelle entfernt wird.

Spezielle Anweisungen und Klauseln

In diesem Abschnitt werden verschiedene Klauseln beschrieben, die zur Laufzeit bereitgestellte SQL-Erweiterungen sind, sowie zwei Sprachelemente, die in vielen Anweisungen, Kommentaren und Ausdrücken verwendet werden können.

COLLATE

Die COLLATE-Klausel wird in SELECT-, CREATE TABLE- und CREATE INDEX-Anweisungen verwendet, um den Vergleichsalgorithmus anzugeben, der beim Vergleichen oder Sortieren von Werten verwendet wird.

```
sql-statement   ::=  COLLATE collation-name
collation-name  ::=  BINARY | NOCASE
```

Der Standardüberprüfungstyp für Spalten ist BINARY. Wenn die BINARY-Überprüfung mit Werten der TEXT-Speicherklasse verwendet wird, wird die binäre Überprüfung ausgeführt, indem die Bytes im Speicher verglichen werden, die den Wert repräsentieren, unabhängig von der Textkodierung.

Die NOCASE-Überprüfungssequenz wird nur für Werte der TEXT-Speicherklasse angewendet. Wenn sie verwendet wird, führt die NOCASE-Überprüfung einen Vergleich durch, bei dem die Groß- und Kleinschreibung nicht beachtet wird.

Für Speicherklassen des Typs NULL, BLOB, INTEGER und REAL wird keine Überprüfungssequenz verwendet.

Um einen anderen Überprüfungstyp als BINARY für eine Spalte zu verwenden, muss eine COLLATE-Klausel als Teil der Spaltendefinition in der CREATE TABLE-Anweisung angegeben werden. Wenn zwei TEXT-Werte verglichen werden, wird eine Überprüfungssequenz verwendet, um die Ergebnisse des Vergleichs gemäß den folgenden Regeln zu bestimmen:

- Wenn bei binären Vergleichsoperatoren einer der Operanden eine Spalte ist, bestimmt der Standardvergleichstyp der Spalte die für den Vergleich zu verwendende Vergleichsreihenfolge. Wenn beide Operanden Spalten sind, bestimmt der Überprüfungstyp für den linken Operanden die verwendete Überprüfungssequenz. Wenn keiner der Operanden eine Spalte ist, wird die BINARY-Überprüfungssequenz verwendet.
- Der Operator BETWEEN...AND entspricht der Verwendung von zwei Ausdrücken mit den Operatoren >= und <=. So entspricht der Ausdruck x BETWEEN y AND z beispielsweise x >= y AND x <= z. Deshalb wird die Vergleichsreihenfolge für den Operator BETWEEN...AND anhand der vorangehenden Regel bestimmt.
- Der IN-Operator verhält sich wie der Operator =, um die zu verwendende Überprüfungssequenz zu bestimmen. Beispielsweise entspricht die Vergleichsreihenfolge für den Ausdruck x IN (y, z) dem Standardvergleichstyp von x, sofern x eine Spalte ist. Andernfalls wird die BINARY-Überprüfung verwendet.
- Einer ORDER BY-Klausel, die Teil einer SELECT-Anweisung ist, kann explizit eine Überprüfungssequenz zugeordnet werden, die für den Sortiervorgang verwendet wird. In diesem Fall wird immer die explizite Überprüfungssequenz verwendet. Wenn dies nicht der Fall ist und wenn der Ausdruck, der von einer ORDER BY-Klausel sortiert wird, eine Spalte ist, richtet sich die Sortierreihenfolge nach dem Standardvergleichstyp der Spalte. Wenn es sich beim Ausdruck nicht um eine Spalte handelt, wird die BINARY-Überprüfungssequenz verwendet.

EXPLAIN

Der EXPLAIN-Befehlsmodifizierer ist eine nicht standardmäßige SQL-Erweiterung.

```
sql-statement  ::=  EXPLAIN sql-statement
```

Wenn das EXPLAIN-Schlüsselwort vor jeder anderen SQL-Anweisung erscheint, wird der Befehl nicht ausgeführt, sondern es wird die Abfolge der Virtual-Machine-Anweisungen angegeben, mit denen der Befehl ausgeführt worden wäre, falls das EXPLAIN-Schlüsselwort nicht vorhanden wäre. Die EXPLAIN-Funktion ist eine erweiterte Funktion und ermöglicht Entwicklern, den SQL-Anweisungstext zu ändern, um zu versuchen, die Leistung zu optimieren, oder um eine Anweisung zu debuggen, die nicht erwartungsgemäß funktioniert.

ON CONFLICT (Konfliktalgorithmen)

Die ON CONFLICT-Klausel ist kein separater SQL-Befehl. Es handelt sich um eine nicht standardmäßige Klausel, die in vielen anderen SQL-Anweisungen verwendet werden kann.

```
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-clause     ::=  OR conflict-algorithm
conflict-algorithm  ::=  ROLLBACK |
                         ABORT |
                         FAIL |
                         IGNORE |
                         REPLACE
```

Die erste Form der ON CONFLICT-Klausel, bei der die ON CONFLICT-Schlüsselwörter verwendet werden, wird in einer CREATE TABLE-Anweisung verwendet. Für eine INSERT- oder UPDATE-Anweisung wird die zweite Form verwendet, wobei ON CONFLICT durch OR ersetzt wird, um der Syntax ein natürlicheres Erscheinungsbild zu geben. Anstelle von INSERT ON CONFLICT IGNORE wird die Anweisung zum Beispiel zu INSERT OR IGNORE. Zwar unterscheiden sich die Schlüsselwörter, die Bedeutung der Klausel ist jedoch in beiden Formen dieselbe.

Die ON CONFLICT-Klausel gibt den Algorithmus an, der zur Auflösung von Beschränkungskonflikten verwendet wird. Die fünf Algorithmen sind ROLLBACK, ABORT, FAIL, IGNORE und REPLACE. Der Standardalgorithmus lautet ABORT. Nachstehend werden die fünf Konfliktalgorithmen erläutert:

ROLLBACK Wenn eine Beschränkungsverletzung auftritt, wird sofort ROLLBACK durchgeführt, wodurch die aktuelle Transaktion beendet wird. Der Befehl wird abgebrochen und die SQLStatement-Instanz löst ein error-Ereignis aus. Wenn keine andere Transaktion aktiv ist (abgesehen von der impliziten Transaktion, die bei jedem Befehl erstellt wird), funktioniert dieser Algorithmus genau wie ABORT.

ABORT Bei einer Beschränkungsverletzung macht der Befehl alle bereits durchgeführten Änderungen rückgängig und die SQLStatement-Instanz löst ein Fehlerereignis aus. Es wird kein ROLLBACK ausgeführt, sodass Änderungen von früheren Befehlen innerhalb einer Transaktion beibehalten werden. ABORT ist das Standardverhalten.

FAIL Wenn eine Beschränkungsverletzung auftritt, wird der Befehl abgebrochen und die SQLStatement-Instanz löst ein Fehlerereignis aus. Änderungen an der Datenbank, die die Anweisung vorgenommen hat, bevor die Beschränkungsverletzung aufgetreten ist, werden jedoch beibehalten und nicht zurückgenommen. Beispiel: Wenn eine UPDATE-Anweisung in Zeile 100 eine Beschränkungsverletzung erkennt, werden die Änderungen der ersten 99 Zeilen beibehalten, aber an Zeile 100 und weiteren Zeilen werden keine Änderungen vorgenommen.

IGNORE Wenn eine Beschränkung verletzt wird, wird die eine Zeile mit der Beschränkungsverletzung nicht eingefügt oder geändert. Abgesehen von dieser Zeile, die ignoriert wird, wird der Befehl wie gewohnt ausgeführt. Andere Zeilen vor und nach der Zeile mit der Beschränkungsverletzung werden normal eingefügt oder aktualisiert. Es wird kein Fehler ausgegeben.

REPLACE Wenn eine UNIQUE-Beschränkungsverletzung auftritt, werden die bereits vorhandenen Zeilen, die die Verletzung verursacht haben, entfernt, bevor die aktuelle Zeile eingefügt oder aktualisiert wird. Das Einfügen oder Aktualisieren findet also immer statt und der Befehl wird normal ausgeführt. Es wird kein Fehler ausgegeben. Wenn eine Verletzung der NOT NULL-Beschränkung auftritt, wird der NULL-Wert durch den Standardwert für diese Spalte ersetzt. Hat die Spalte keinen Standardwert, wird der ABORT-Algorithmus verwendet. Wenn die CHECK-Beschränkung verletzt wird, wird der IGNORE-Algorithmus verwendet. Wenn diese Strategie zur Konfliktlösung Zeilen löscht, um eine Beschränkung zu erfüllen, werden keine delete-Auslöser für diese Zeilen aufgerufen.

Der in der OR-Klausel einer INSERT- oder UPDATE-Anweisung angegebene Algorithmus hat Vorrang vor allen Algorithmen, die in einer CREATE TABLE-Anweisung angegeben sind. Wenn in der CREATE TABLE-Anweisung oder in der ausgeführten INSERT- oder UPDATE-Anweisung kein Algorithmus angegeben ist, wird der ABORT-Algorithmus verwendet.

REINDEX

Mit dem REINDEX-Befehl werden Indizes gelöscht und neu erstellt. Dieser Befehl ist hilfreich, wenn die Definition einer Überprüfungssequenz geändert wurde.

```
sql-statement  ::=  REINDEX collation-name
sql-statement  ::=  REINDEX [database-name .] ( table-name | index-name )
```

In der ersten Form werden alle Indizes in allen verbundenen Datenbanken, die die genannte Überprüfungssequenz verwenden, neu erstellt. In der zweiten Form werden alle der Tabelle zugeordneten Indizes neu erstellt, wenn ein table-name angegeben wird. Wenn ein index-name angegeben wird, wird nur diese Indexposition gelöscht und neu erstellt.

KOMMENTARE

Kommentare sind keine SQL-Befehle, können aber in SQL-Abfragen vorkommen. Sie werden von der Laufzeitumgebung wie Leerraum behandelt. Kommentare können überall dort beginnen, wo Leerraum zu finden ist, auch innerhalb von Ausdrücken, die mehrere Zeilen umfassen.

```
comment             ::=  single-line-comment |
                         block-comment
single-line-comment ::=  -- single-line
block-comment       ::=  /* multiple-lines or block [*/]
```

Einzeilige Kommentare sind durch zwei Bindestriche gekennzeichnet. Ein einzeiliger Kommentar geht nicht über das Ende der aktuellen Zeile hinaus.

Kommentarblöcke können beliebig viele Zeilen umfassen oder sich in einer einzelnen Zeile befinden. Wenn kein abschließendes Begrenzungszeichen vorhanden ist, reicht der Kommentar bis zum Ende der Eingabe. Diese Situation wird nicht als Fehler behandelt. Eine neue SQL-Anweisung kann in einer Zeile nach dem Ende eines Kommentarblocks beginnen. Kommentarblöcke können überall eingebettet werden, wo Leerraum auftreten kann, auch innerhalb von Ausdrücken und in anderen SQL-Anweisungen. Kommentarblöcke sind nicht verschachtelt. Einzeilige Kommentare innerhalb eines Kommentarblocks werden ignoriert.

AUSDRÜCKE

Ausdrücke sind Unterbefehle innerhalb anderer SQL-Blöcke. Nachstehend wird die gültige Syntax für einen Ausdruck innerhalb einer SQL-Anweisung beschrieben:

```
expr            ::=  expr binary-op expr |
                     expr [NOT] like-op expr [ESCAPE expr] |
                     unary-op expr |
                     ( expr ) |
                     column-name |
                     table-name.column-name |
                     database-name.table-name.column-name |
                     literal-value |
                     parameter |
                     function-name( expr-list | * ) |
                     expr ISNULL |
                     expr NOTNULL |
                     expr [NOT] BETWEEN expr AND expr |
                     expr [NOT] IN ( value-list ) |
                     expr [NOT] IN ( select-statement ) |
                     expr [NOT] IN [database-name.] table-name |
                     [EXISTS] ( select-statement ) |
                     CASE [expr] ( WHEN expr THEN expr )+ [ELSE expr] END |
                     CAST ( expr AS type ) |
                     expr COLLATE collation-name
like-op         ::=  LIKE | GLOB
binary-op       ::=  see Operators
unary-op        ::=  see Operators
parameter       ::=  :param-name | @param-name | ?
value-list      ::=  literal-value [, literal-value]*
literal-value   ::=  literal-string | literal-number | literal-boolean | literal-blob |
literal-null
literal-string  ::=  'string value'
literal-number  ::=  integer | number
literal-boolean  ::=  true | false
literal-blob  ::=  X'string of hexadecimal data'
literal-null  ::=  NULL
```

Ein Ausdruck ist eine beliebige Kombination aus Werten und Operatoren, die in einen einzelnen Wert aufgelöst werden können. Ausdrücke lassen sich in zwei Grundtypen unterteilen, je nachdem, ob sie in einen booleschen Wert (true oder false) oder in einen nicht booleschen Wert aufgelöst werden.

In verschiedenen häufig vorkommenden Situationen muss der Ausdruck in einen booleschen Wert aufgelöst werden, zum Beispiel in einer WHERE-Klausel, einer HAVING-Klausel, im ON-Ausdruck in einer JOIN-Klausel und in einem CHECK-Ausdruck. Die folgenden Ausdruckstypen erfüllen diese Bedingung:

- ISNULL
- NOTNULL
- IN ()
- EXISTS ()
- LIKE
- GLOB
- Bestimmte Funktionen
- Bestimmte Operatoren (insbesondere Vergleichsoperatoren)

Literalwerte

Ein numerischer Literalwert wird als Ganzzahl oder als Gleitkommazahl geschrieben. Die wissenschaftliche Schreibweise wird unterstützt. Der . (Punkt) wird immer als Dezimaltrennzeichen verwendet.

Ein Stringliteral ist dadurch gekennzeichnet, dass der String in einfache Anführungszeichen ' gesetzt wird. Wenn ein String ein einfaches Anführungszeichen enthält, schreiben Sie zwei einfache Anführungszeichen hintereinander: ''.

Ein boolesches Literal ist durch den Wert true oder false gekennzeichnet. Boolesche Literalwerte werden mit dem booleschen Spaltendatentyp verwendet.

Ein BLOB-Literal ist ein Stringliteral, das hexadezimale Daten enthält und dem ein einzelnes x- oder X-Zeichen vorangestellt ist, zum Beispiel X'53514697465'.

Ein Literalwert kann auch das Token NULL sein.

Spaltenname

Ein Spaltenname kann einer der Namen sein, die in der CREATE TABLE-Anweisung definiert sind, oder einer der folgenden besonderen Bezeichner: ROWID, OID oder _ROWID_. Alle diese besonderen Bezeichner beschreiben den eindeutigen, zufälligen Ganzzahlschlüssel (den „Zeilenschlüssel“), der jeder Zeile einer jeden Tabelle zugeordnet ist. Die speziellen Bezeichner verweisen nur auf den Zeilenschlüssel, wenn die CREATE TABLE-Anweisung keine reale Spalte mit demselben Namen definiert. Zeilenschlüssel verhalten sich wie schreibgeschützte Spalten. Ein Zeilenschlüssel kann überall verwendet werden, wo auch eine reguläre Spalte verwendet werden kann, allerdings kann der Wert eines Zeilenschlüssels in einer UPDATE- oder INSERT-Anweisung nicht geändert werden. Die SELECT * FROM table-Anweisung schließt den Zeilenschlüssel nicht in ihren Ergebnissatz ein.

SELECT-Anweisung

Eine SELECT-Anweisung kann in einem Ausdruck als rechter Operand des IN-Operators, als Skalarmenge (ein einzelner Ergebniswert) oder als Operand eines EXISTS-Operators verwendet werden. Bei der Verwendung als Skalarmenge oder als Operand eines IN-Operators kann die SELECT-Anweisung nur eine einzelne Spalte in ihrem Ergebnis aufweisen. Eine zusammengesetzte SELECT-Anweisung (durch Schlüsselwörter wie UNION oder EXCEPT verbunden) ist zulässig. Mit dem EXISTS-Operator werden die Spalten im Ergebnissatz der SELECT-Anweisung ignoriert und der Ausdruck gibt TRUE zurück, wenn eine oder mehrere Zeilen vorhanden sind, bzw. FALSE, wenn der Ergebnissatz leer ist. Wenn kein Begriff im SELECT-Ausdruck auf den Wert in der enthaltenden Abfrage verweist, wird der Ausdruck einmal evaluiert, bevor eine weitere Verarbeitung erfolgt, und das Ergebnis wird wie erforderlich wiederverwendet. Wenn der SELECT-Ausdruck Variablen aus der äußeren Abfrage enthält (als korrelierte Unterabfrage bezeichnet), wird die SELECT-Anweisung jedes Mal, wenn sie benötigt wird, erneut evaluiert.

Wenn eine SELECT-Anweisung der rechte Operand des IN-Operator ist, gibt der IN-Operator den Wert TRUE zurück, wenn das Ergebnis des linken Operanden gleich einem der Werte im Ergebnissatz der SELECT-Anweisung ist. Dem IN-Operator kann das NOT-Schlüsselwort vorangestellt werden, um den Sinn des Tests umzukehren.

Wenn eine SELECT-Anweisung innerhalb eines Ausdrucks auftritt, aber nicht der rechte Operand eines IN-Operators ist, wird die erste Zeile im Ergebnis der SELECT-Anweisung als Wert im Ausdruck verwendet. Wenn die SELECT-Anweisung mehr als eine Ergebniszeile erzielt, werden alle Zeilen nach der ersten ignoriert. Wenn die SELECT-Anweisung keine Zeilen ergibt, hat die SELECT-Anweisung den Wert NULL.

CAST-Ausdruck

Ein CAST-Ausdruck ändert den Datentyp des angegebenen Werts zu dem gegebenen. Der angegebene Typ kann jeder nicht leere Typname sein, der für den Typ in einer Spaltendefinition einer CREATE TABLE-Anweisung gültig ist. Ausführliche Informationen finden Sie unter „Unterstützte Datentypen“.

Weitere Ausdruckselemente

Die folgenden SQL-Elemente können ebenfalls in Ausdrücken verwendet werden:

- Integrierte Funktionen: Aggregatfunktionen, Skalarfunktionen und Funktionen zur Formatierung von Datum und Uhrzeit
- Operatoren
- Parameter

Integrierte Funktionen

Die integrierten Funktionen lassen sich in drei Hauptkategorien aufteilen:

- Aggregationsfunktionen
- Skalarfunktionen
- Datums- und Uhrzeitfunktionen

Zusätzlich zu diesen Funktionen gibt es eine besondere Funktion, RAISE(), mit der eine Benachrichtigung ausgegeben wird, wenn beim Ausführen eines Auslösers ein Fehler auftritt. Diese Funktion kann nur im Hauptteil einer CREATE TRIGGER-Anweisung verwendet werden. Informationen zur RAISE()-Funktion finden Sie unter CREATE TRIGGER > RAISE().

Wie bei allen Schlüsselwörtern in SQL spielt die Groß- und Kleinschreibung bei Funktionsnamen keine Rolle.

Aggregationsfunktionen

Aggregationsfunktionen führen Operationen an Werten aus mehreren Zeilen aus. Diese Funktionen werden hauptsächlich in SELECT-Anweisungen zusammen mit der GROUP BY-Klausel verwendet.

AVG(X)	Gibt den Durchschnittswert aller X-Werte ungleich NULL innerhalb einer Gruppe zurück. String- und BLOB- Werte, die nicht wie Zahlen aussehen, werden als 0 interpretiert. Das Ergebnis von AVG() ist immer ein Gleitkommawert, selbst wenn alle Eingaben Ganzzahlen sind.
COUNT(X) COUNT(*)	Die Rückgabe der ersten Form gibt an, wie oft X in einer Gruppe nicht NULL ist. Die zweite Form (mit dem Argument *) gibt die Gesamtzahl der Zeilen in der Gruppe zurück.
MAX(X)	Gibt den Höchstwert aller Werte in der Gruppe zurück. Es wird die übliche Sortierreihenfolge verwendet, um den Höchstwert zu bestimmen.
MIN(X)	Gibt den kleinsten Wert aller Werte ungleich NULL in der Gruppe zurück. Es wird die übliche Sortierreihenfolge verwendet, um den niedrigsten Wert zu bestimmen. Wenn alle Werte in der Gruppe NULL sind, wird NULL zurückgegeben.
SUM(X) TOTAL(X)	Gibt die numerische Summe aller Werte ungleich NULL in der Gruppe zurück. Wenn alle Werte NULL sind, hat SUM() den Rückgabewert NULL und TOTAL() gibt 0.0 zurück. Das Ergebnis von TOTAL() ist immer ein Gleitpunktwert. Das Ergebnis von SUM() ist ein ganzzahliger Wert, wenn alle Eingaben ungleich NULL Ganzzahlen sind. Wenn eine Eingabe für SUM() keine Ganzzahl und nicht NULL ist, gibt SUM() einen Gleitkommawert zurück. Dieser Wert kann eine Annäherung an die tatsächliche Summe sein.

In jeder der vorstehenden Aggregationsfunktionen, die ein einzelnes Argument verwenden, kann diesem Argument das DISTINCT-Schlüsselwort vorangestellt werden. In diesem Fall werden doppelte Elemente herausgefiltert, bevor sie an die Aggregationsfunktion übergeben werden. So gibt der Funktionsaufruf COUNT(DISTINCT x) beispielsweise die Anzahl der unterschiedlichen Werte in Spalte X anstelle der Gesamtanzahl der Werte ungleich NULL in Spalte X zurück.

Skalarfunktionen

Skalarfunktionen werden jeweils für die Werte einer Zeile ausgeführt.

ABS(X)	Gibt den absoluten Wert des Arguments X zurück.

COALESCE(X, Y, ...)	Gibt eine Kopie des ersten Arguments zurück, das nicht NULL ist. Wenn alle Argumente NULL sind, wird NULL zurückgegeben. Es müssen mindestens zwei Argumente vorhanden sein.
GLOB(X, Y)	Mit dieser Funktion wird die X GLOB Y-Syntax implementiert.
IFNULL(X, Y)	Gibt eine Kopie des ersten Arguments zurück, das nicht NULL ist. Wenn beide Argumente NULL sind, wird NULL zurückgegeben. Diese Funktion verhält sich wie COALESCE().
HEX(X)	Das Argument wird als Wert des BLOB-Speichertyps interpretiert. Das Ergebnis ist eine hexadezimale Darstellung des Inhalts dieses Werts.
LAST_INSERT_ROWID( )	Gibt den Zeilenbezeichner (generierter Primärschlüssel) der letzten Zeile zurück, die über die aktuelle SQLConnection-Instanz eingefügt wurde. Der Wert entspricht dem Wert, der von der SQLConnection.lastInsertRowID-Eigenschaft zurückgegeben wird.
LENGTH(X)	Gibt die Stringlänge von X in Zeichen zurück.
LIKE(X, Y [, Z])	Mit dieser Funktion wird die X LIKE Y [ESCAPE Z]-Syntax implementiert. Wenn die optionale ESCAPE-Klausel vorhanden ist, wird die Funktion mit drei Argumenten aufgerufen. Andernfalls wird sie nur mit zwei Argumenten aufgerufen.
LOWER(X)	Gibt eine Kopie des Strings X zurück, wobei alle Zeichen in Kleinschreibung konvertiert werden.
LTRIM(X) LTRIM(X, Y)	Gibt einen String zurück, der gebildet wird, indem Leerzeichen links von X entfernt werden. Wenn ein Y-Argument angegeben wird, entfernt die Funktion die Zeichen in Y links von X.
MAX(X, Y, ...)	Gibt das Argument mit dem Höchstwert zurück. Argumente können neben Zahlen auch Strings sein. Der Höchstwert wird von der definierten Sortierreihenfolge bestimmt. Beachten Sie, dass MAX() eine einfache Funktion ist, wenn sie mindestens zwei Argumente hat, aber eine Aggregationsfunktion, wenn sie nur ein Argument hat.
MIN(X, Y, ...)	Gibt das Argument mit dem niedrigsten Wert zurück. Argumente können neben Zahlen auch Strings sein. Der niedrigste Wert wird von der definierten Sortierreihenfolge bestimmt. Beachten Sie, dass MIN() eine einfache Funktion ist, wenn sie mindestens zwei Argumente hat, aber eine Aggregationsfunktion, wenn sie nur ein Argument hat.
NULLIF(X, Y)	Gibt das erste Argument zurück, wenn die Argumente sich unterscheiden, andernfalls wird NULL zurückgegeben.
QUOTE(X)	Diese Routine gibt einen String zurück, der dem Wert ihres Arguments entspricht, der in eine andere SQL-Anweisung eingeschlossen werden kann. Strings sind in einfache Anführungszeichen gesetzt. Wenn innerhalb des Strings Anführungszeichen erforderlich sind, werden Escape-Zeichen vorangestellt. BLOB-Speicherklassen werden als hexadezimale Literale kodiert. Diese Funktion ist hilfreich, wenn Auslöser geschrieben werden, um Rückgängig-/Wiederherstellen-Funktionen zu implementieren.
RANDOM(*)	Gibt eine pseudo-zufällige Ganzzahl zwischen -9223372036854775808 und 9223372036854775807 zurück. Dieser Zufallswert ist nicht entschlüsselungsresistent.
RANDOMBLOB(N)	Gibt einen BLOB-Wert mit N Byte zurück, der pseudozufällige Byte enthält. N sollte eine positive Ganzzahl sein. Dieser Zufallswert ist nicht entschlüsselungsresistent. Wenn der Wert von N negativ ist, wird ein einzelnes Byte zurückgegeben.
ROUND(X) ROUND(X, Y)	Rundet die Zahl X auf Y Ziffern rechts vom Dezimaltrennzeichen ab. Wenn das Argument Y ausgelassen wird, wird 0 verwendet.
RTRIM(X) RTRIM(X, Y)	Gibt einen String zurück, der gebildet wird, indem Leerzeichen rechts von X entfernt werden. Wenn ein Y-Argument angegeben wird, entfernt die Funktion die Zeichen in Y rechts von X.
SUBSTR(X, Y, Z)	Gibt einen Teilstring des Eingabestrings X zurück, der mit dem Y. Zeichen beginnt und Z Zeichen lang ist. Das Zeichen ganz links von X hat die Indexposition 1. Wenn Y negativ ist, wird das erste Zeichen des Teilstrings ermittelt, indem von rechts anstelle von links gezählt wird.
TRIM(X) TRIM(X, Y)	Gibt einen String zurück, der gebildet wird, indem Leerzeichen rechts von X entfernt werden. Wenn ein Y-Argument angegeben wird, entfernt die Funktion die Zeichen in Y rechts von X.
TYPEOF(X)	Gibt den Typ des Ausdrucks X zurück. Die möglichen Rückgabewerte sind „null“, „integer“, „real“, „text“ und „blob“. Weitere Informationen zu Datentypen finden Sie unter „Unterstützte Datentypen“.
UPPER(X)	Gibt eine Kopie des Eingabestrings X zurück, wobei alle Buchstaben in Großbuchstaben umgewandelt werden.
ZEROBLOB(N)	Gibt einen BLOB mit N Byte von 0x00 zurück.

Datums-/Uhrzeitformat-Funktionen

Die Funktionen zur Formatierung von Datum und Uhrzeit sind eine Gruppe von Skalarfunktionen, mit denen formatierte Datums- und Uhrzeitdaten erstellt werden. Beachten Sie, dass diese Funktionen für String- und Zahlenwerte ausgeführt werden und String- und Zahlenwerte zurückgeben. Diese Funktionen sind nicht für die Verwendung mit dem DATE-Datentyp konzipiert. Wenn Sie diese Funktionen für Daten in einer Spalte verwenden, deren deklarierter Datentyp DATE ist, verhalten sie sich nicht wie erwartet.

DATE(T, ...)	Die DATE()-Funktion gibt einen String zurück, der das Datum im folgenden Format enthält: YYYY-MM-DD. Der erste Parameter (T) gibt einen Zeitstring in einem Format zurück, das unter „Zeitformate“ zu finden ist. Nach dem Zeitstring kann eine beliebige Anzahl von Modifizierern angegeben werden. Die Modifizierer sind unter „Modifizierer“ aufgeführt.
TIME(T, ...)	Die TIME()-Funktion gibt einen String zurück, der die Uhrzeit im Format HH:MM:SS enthält. Der erste Parameter (T) gibt einen Zeitstring in einem Format zurück, das unter „Zeitformate“ zu finden ist. Nach dem Zeitstring kann eine beliebige Anzahl von Modifizierern angegeben werden. Die Modifizierer sind unter „Modifizierer“ aufgeführt.
DATETIME(T, ...)	Die DATETIME()-Funktion gibt einen String zurück, der Datum und Uhrzeit im Format YYYY-MM-DD HH:MM:SS enthält. Der erste Parameter (T) gibt einen Zeitstring in einem Format zurück, das unter „Zeitformate“ zu finden ist. Nach dem Zeitstring kann eine beliebige Anzahl von Modifizierern angegeben werden. Die Modifizierer sind unter „Modifizierer“ aufgeführt.
JULIANDAY(T, ...)	Die JULIANDAY()-Funktion gibt eine Zahl zurück, die die Anzahl der Tage seit Mittag in Greenwich am 24. November 4714 v. Chr. und das bereitgestellte Datum angibt. Der erste Parameter (T) gibt einen Zeitstring in einem Format zurück, das unter „Zeitformate“ zu finden ist. Nach dem Zeitstring kann eine beliebige Anzahl von Modifizierern angegeben werden. Die Modifizierer sind unter „Modifizierer“ aufgeführt.
STRFTIME(F, T, ...)	Die STRFTIME()-Routine gibt das Datum zurück, das anhand des Formatstrings formatiert ist, der als das erste Argument F angegeben ist. Der Formatstring kann durch folgende Angaben ersetzt werden: %d – Monatstag %f – Sekunden als Bruch SS.SSS %H – Stunde 00-24 %j – Tag des Jahres 001-366 %J – Tag in julianischer Angabe %m – Monat 01-12 %M – Minute 00-59 %s – Sekunden seit 01.01.1970 %S – Sekunden 00-59 %w – Wochentag 0-6 (Sonntag = 0) %W – Jahreswoche 00-53 %Y – Jahr 0000-9999 %% - % Der zweite Parameter (T) gibt einen Zeitstring in einem Format zurück, das unter „Zeitformate“ zu finden ist. Nach dem Zeitstring kann eine beliebige Anzahl von Modifizierern angegeben werden. Die Modifizierer sind unter „Modifizierer“ aufgeführt.

Zeitformate

Ein Zeitstring kann in einem der folgenden Formate angegeben werden:

YYYY-MM-DD.	2007-06-15
YYYY-MM-DD HH:MM	2007-06-15 07:30
YYYY-MM-DD HH:MM:SS	2007-06-15 07:30:59
YYYY-MM-DD HH:MM:SS.SSS	2007-06-15 07:30:59.152
YYYY-MM-DDTHH:MM	2007-06-15T07:30

YYYY-MM-DDTHH:MM:SS	2007-06-15T07:30:59
YYYY-MM-DDTHH:MM:SS.SSS	2007-06-15T07:30:59.152
HH:MM	07:30 (Datum ist 2000-01-01)
HH:MM:SS	07:30:59 (Datum ist 2000-01-01)
HH:MM:SS.SSS	07:30:59:152 (Datum ist 2000-01-01)
now	Das aktuelle Datum und die Uhrzeit als Universal Coordinated Time (UTC).
DDDD.DDDD	Tag in julianischer Angabe als Gleitpunktzahl.

Das Zeichen T in diesen Formaten ist ein Literalzeichen "T", das Datum und Uhrzeit trennt. Formate, die nur die Uhrzeit enthalten, gehen vom Datum 2001-01-01 aus.

Modifizierer

Dem Zeitstring können null oder mehr Modifizierer nachgestellt werden, die das Datum oder die Interpretation des Datums ändern. Folgende Modifizierer stehen zur Verfügung:

NNN days	Anzahl der Tage, die der Zeit hinzugefügt werden sollen.
NNN hours	Anzahl der Stunden, die der Zeit hinzugefügt werden sollen.
NNN minutes	Anzahl der Minuten, die der Zeit hinzugefügt werden sollen.
NNN.NNNN seconds	Anzahl der Sekunden und Millisekunden, die der Zeit hinzugefügt werden sollen.
NNN months	Anzahl der Monate, die der Zeit hinzugefügt werden sollen.
NNN years	Anzahl der Jahre, die der Zeit hinzugefügt werden sollen.
start of month	Versetzt die Zeit rückwärts bis zum Anfang des Monats.
start of year	Versetzt die Zeit rückwärts bis zum Anfang des Jahres.
start of day	Versetzt die Zeit rückwärts bis zum Anfang des Tages.
weekday N	Versetzt die Zeit vorwärts bis zum angegebenen Wochentag. (0 = Sonntag, 1 = Montag usw.).
localtime	Konvertiert das Datum in die lokale Zeit.
utc	Konvertiert das Datum in die Weltzeit (UCT).

Operatoren

SQL unterstützt eine große Auswahl von Operatoren, darunter gebräuchliche Operatoren, die in den meisten Programmiersprachen vorkommen, aber auch verschiedene Operatoren, die nur in SQL verwendet werden.

Gebräuchliche Operatoren

Die folgenden binären Operatoren sind in einem SQL-Block zulässig und werden hier von der höchsten zur niedrigsten Priorität aufgeführt:

```
*     /     %
+     -
<<  >>  &  |
<  >=    >  >=
=     ==    !=    <>  IN
AND
OR
```

Unterstützte unäre Präfix-Operatoren:

```
 !     ~     NOT
```

Den COLLATE-Operator kann man sich als unären Postfix-Operator vorstellen. Der COLLATE-Operator hat die höchste Priorität. Er stellt immer eine festere Bindung her als jeder unäre Präfix-Operator und jeder binäre Operator.

Beachten Sie, dass es zwei Variationen der Gleichheits- und Ungleichheitsoperatoren gibt. „Ist gleich“ kann = oder == sein. „Ist nicht gleich“ kann != oder <> sein.

|| ist ein Stringverkettungsoperator – er verbindet die beiden Strings seiner Operanden.

Der Operator % gibt den Rest der Division des linken Operanden durch den rechten Operanden zurück.

Das Ergebnis jedes binären Operators ist ein numerischer Wert, mit Ausnahme des Verkettungsoperators ||, der ein Stringergebnis liefert.

SQL-Operatoren

LIKE

Der LIKE-Operator führt einen Mustervergleich aus.

```
expr     ::=  (column-name | expr) LIKE pattern
pattern  ::=  '[ string | % | _ ]'
```

Der Operand rechts vom LIKE-Operator enthält das Muster und der Operator auf der linken Seite enthält den String, der mit dem Muster verglichen wird. Das Prozentzeichen (%) im Muster ist ein Platzhalter, der einer beliebigen Folge aus null oder mehr Zeichen im String entspricht. Ein Unterstrich (_) im Muster stimmt mit einem beliebigen einzelnen Zeichen im String überein. Jedes andere Zeichen stimmt mit sich selbst bzw. mit seinem Äquivalent in Groß-/Kleinschreibung überein, da die Groß- oder Kleinschreibung keine Rolle spielt. (Hinweis: Die Datenbankengine kann Groß-/Kleinbuchstaben nur für lateinische 7-Bit-Zeichen interpretieren. Dementsprechend beachtet der LIKE-Operator die Groß-/Kleinschreibung für 8-Bit-ISO8859-Zeichen oder UTF-Zeichen. Zum Beispiel: Der Ausdruck 'a' LIKE 'A' ist TRUE, aber 'æ' LIKE 'Æ' ist FALSE). Die Beachtung der Groß-/Kleinschreibung kann für lateinische Buchstaben mit der SQLConnection.caseSensitiveLike-Eigenschaft geändert werden.

Wenn die optionale ESCAPE-Klausel vorhanden ist, muss der Ausdruck nach dem ESCAPE-Schlüsselwort in einen String evaluiert werden, der aus einem einzelnen Zeichen besteht. Dieses Zeichen kann im LIKE-Muster verwendet werden, um Prozent- oder Unterstrich-Literalzeichen zu finden. Das Escape-Zeichen, dem ein Prozentsymbol, ein Unterstrich oder ein Escape-Zeichen folgt, stimmt mit einem tatsächlichen Prozentsymbol, Unterstrich bzw. Escape-Zeichen im String überein.

GLOB

Der GLOB-Operator ähnelt LIKE, verwendet für Platzhalter jedoch die Unix-Syntax für das automatische Vervollständigen von Dateien (File Globbing). Anders als LIKE beachtet GLOB die Groß- und Kleinschreibung.

IN

Der IN-Operator berechnet, ob sein linker Operand gleich einem der Werte im rechten Operanden (ein Satz von Werten in Klammern) ist.

```
in-expr         ::=  expr [NOT] IN ( value-list ) |
                     expr [NOT] IN ( select-statement ) |
                     expr [NOT] IN [database-name.] table-name
value-list      ::=  literal-value [, literal-value]*
```

Der rechte Operand kann ein Satz von kommagetrennten Literalwerten oder das Ergebnis einer SELECT-Anweisung sein. Im Abschnitt zur SELECT-Anweisung finden Sie Erläuterungen sowie Informationen zu Beschränkungen bei der Verwendung von SELECT-Anweisung als rechter Operand des IN-Operators.

BETWEEN...AND

Der Operator BETWEEN...AND entspricht der Verwendung von zwei Ausdrücken mit den Operatoren >= und <=. So entspricht der Ausdruck x BETWEEN y AND z beispielsweise x > y AND x <= z.

NOT

Der NOT-Operator ist ein Negationsoperator. Den Operatoren GLOB, LIKE und IN kann das NOT-Schlüsselwort vorangestellt sein, um den Sinn des Tests in sein Gegenteil umzukehren (anders ausgedrückt, um zu prüfen, dass ein Wert nicht dem angegebenen Muster entspricht).

Parameter

Ein Parameter gibt im Ausdruck einen Platzhalter für einen Literalwert an, der zur Laufzeit gefüllt wird, indem dem assoziativen SQLStatement.parameters-Array ein Wert zugewiesen wird. Parameter können drei Formen annehmen:

?	Ein Fragezeichen gibt einen indizierten Parameter an. Parametern werden entsprechend ihrer Reihenfolge in der Anweisung numerische (auf null basierende) Indexwerte zugewiesen.
:AAAA	Ein Doppelpunkt gefolgt von einem Bezeichnernamen ist ein Platzhalter für einen benannten Parameter mit dem Namen AAAA. Benannte Parameter sind ebenfalls entsprechend ihrer Reihenfolge in der SQL-Anweisung nummeriert. Der Übersichtlichkeit halber sollten benannte und nummerierte Parameter nicht gemischt werden.
@AAAA	Das @-Zeichen entspricht einem Doppelpunkt.

Nicht unterstützte SQL-Funktionen

In der folgenden Liste sind die SQL-Standardelemente aufgeführt, die von Adobe AIR nicht unterstützt werden:

FOREIGN KEY-Beschränkungen FOREIGN KEY-Beschränkungen werden analysiert, aber nicht erzwungen.

Auslöser Auslöser des Typs FOR EACH STATEMENT werden nicht unterstützt (alle Auslöser müssen den Typ FOR EACH ROW haben). INSTEAD OF-Auslöser werden für Tabellen nicht unterstützt (INSTEAD OF-Auslöser sind nur für Ansichten zulässig). Rekursive Auslöser (d. h. Auslöser, die sich selbst auslösen) werden nicht unterstützt.

ALTER TABLE Nur die Varianten RENAME TABLE und ADD COLUMN des ALTER TABLE-Befehls werden unterstützt. Andere Arten von ALTER TABLE-Operationen, zum Beispiel DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT usw., werden ignoriert.

Verschachtelte Transaktionen Nur eine aktive Transaktion ist zulässig.

RIGHT und FULL OUTER JOIN RIGHT OUTER JOIN oder FULL OUTER JOIN werden nicht unterstützt.

Aktualisierbare Ansicht (VIEW) Eine Ansicht ist schreibgeschützt. Sie können keine DELETE-, INSERT- oder UPDATE-Anweisung für eine Ansicht ausführen. Ein INSTEAD OF-Auslöser, der beim Versuch eines DELETE-, INSERT- oder UPDATE-Vorgangs für eine Ansicht ausgelöst wird, wird unterstützt und kann verwendet werden, um unterstützende Tabellen im Hauptteil des Auslösers zu aktualisieren.

GRANT und REVOKE Eine Datenbank ist eine reguläre Festplattendatei, auf die nur die normalen Dateizugriffsberechtigungen des zugrunde liegenden Betriebssystems angewendet werden können. Die Befehle GRANT und REVOKE, die häufig in Client/Server-RDBMS verwendet werden, werden nicht implementiert.

Die folgenden SQL-Elemente und SQLite-Funktionen werden in einigen SQLite-Implementierungen unterstützt, jedoch nicht von Adobe AIR. Der Großteil dieser Funktionalität ist über Methoden der SQLConnection-Klasse verfügbar:

Transaktionsbezogene SQL-Elemente (BEGIN, END, COMMIT, ROLLBACK) Diese Funktionalität steht über die transaktionsbezogenen Methoden der SQLConnection-Klasse zur Verfügung: SQLConnection.begin(), SQLConnection.commit() und SQLConnection.rollback().

ANALYZE Diese Funktionalität steht mit der SQLConnection.analyze()-Methode zur Verfügung.

ATTACH Diese Funktionalität steht mit der SQLConnection.attach()-Methode zur Verfügung.

COPY Diese Anweisung wird nicht unterstützt.

CREATE VIRTUAL TABLE Diese Anweisung wird nicht unterstützt.

DETACH Diese Funktionalität steht mit der SQLConnection.detach()-Methode zur Verfügung.

PRAGMA Diese Anweisung wird nicht unterstützt.

VACUUM Diese Funktionalität steht mit der SQLConnection.compact()-Methode zur Verfügung.

Zugriff auf Systemtabellen nicht verfügbar Die Systemtabellen, einschließlich sqlite_master und anderen Tabellen mit dem Präfix "sqlite_", sind in SQL-Anweisungen nicht verfügbar. Die Laufzeitumgebung enthält eine Schema-API, die den objektorientierten Zugriff auf Schemadaten ermöglicht. Weitere Informationen hierzu finden Sie im Abschnitt zur SQLConnection.loadSchema()-Methode.

Funktionen für reguläre Ausdrücke (MATCH() und REGEX()) Diese Funktionen stehen in SQL-Anweisungen nicht zur Verfügung.

Die folgende Funktionalität unterscheidet sich zwischen vielen SQLite-Implementierungen und Adobe AIR:

Indizierte Anweisungsparameter In vielen Implementierungen sind indizierte Anweisungsparameter eins-basiert. In Adobe AIR sind indizierte Anweisungsparameter jedoch null-basiert (der erste Parameter hat den Index 0, der zweite hat den Index 1 usw.).

INTEGER PRIMARY KEY-Spaltendefinitionen Bei vielen Implementierungen werden nur Spalten, die genau als INTEGER PRIMARY KEY definiert sind, als tatsächlicher Primärschlüssel der Tabelle verwendet. Werden in solchen Implementierungen andere Datentypen synonym für INTEGER verwendet (z. B. int), wird die betreffende Spalte nicht als interner Primärschlüssel verwendet. In Adobe AIR gilt jedoch der int-Datentyp (sowie andere Synonyme für INTEGER) als genau äquivalent mit INTEGER. Demzufolge wird eine als int PRIMARY KEY definierte Spalte als interner Primärschlüssel für die Tabelle verwendet. Weitere Informationen finden Sie unter CREATE TABLE und im Abschnitt zur Spaltenaffinität.

Zusätzliche SQL-Funktionen

Die folgenden Spaltenaffinitätstypen werden standardmäßig nicht von SQLite unterstützt, jedoch in Adobe AIR (beachten Sie, dass bei diesen Datentypen die Groß- und Kleinschreibung wie bei allen Schlüsselwörtern in SQL nicht wichtig ist):

Boolean entspricht der Boolean-Klasse.

Date entspricht der Date-Klasse.

int entspricht der int-Klasse (äquivalent zur INTEGER-Spaltenaffinität).

Number entspricht der Number-Klasse (äquivalent zur REAL-Spaltenaffinität).

Object entspricht der Object-Klasse oder einer beliebigen Unterklasse, die mit AMF3 serialisiert und deserialisiert werden kann. (Dazu gehören die meisten Klassen, wie zum Beispiel benutzerdefinierte Klassen; einige Klassen, darunter Anzeigeobjekte und Objekte, die Anzeigeobjekte als Eigenschaften enthalten, sind jedoch ausgeschlossen.)

String entspricht der String-Klasse (äquivalent zur TEXT-Spaltenaffinität).

XML entspricht der ActionScript (E4X) XML-Klasse.

XMLList entspricht der ActionScript (E4X) XMLList-Klasse.

Die folgenden Literalwerte werden standardmäßig in SQLite nicht unterstützt, werden jedoch in Adobe AIR unterstützt:

true repräsentiert den booleschen Literalwert true für die Arbeit mit BOOLEAN-Spalten.

false repräsentiert den booleschen Literalwert false für die Arbeit mit BOOLEAN-Spalten.

Unterstützte Datentypen

Anders als die meisten SQL-Datenbanken erfordert oder erzwingt die SQL-Datenbank-Engine von Adobe AIR nicht, das Tabellenspalten Werte eines bestimmten Typs enthalten. Stattdessen verwendet die Laufzeitumgebung zwei Konzepte, Speicherklassen und Spaltenaffinität, um Datentypen zu steuern. In diesem Abschnitt werden Speicherklassen und Spaltenaffinität beschrieben. Außerdem wird die Auflösung von Datentypunterschieden unter verschiedenen Bedingungen erläutert:

- „Speicherklassen“ auf Seite 1209
- „Spaltenaffinität“ auf Seite 1210
- „Datentypen und Vergleichsoperatoren“ auf Seite 1212
- „Datentypen und mathematische Operatoren“ auf Seite 1213
- „Datentypen und Sortierung“ auf Seite 1213
- „Datentypen und Gruppierung“ auf Seite 1213
- „Datentypen und zusammengesetzte SELECT-Anweisungen“ auf Seite 1214

Speicherklassen

Speicherklassen repräsentieren die aktuellen Datentypen, die zum Speichern von Werten in einer Datenbank verwendet werden. Die folgenden Speicherklassen werden von der Datenbank verwendet:

NULL Der Wert ist ein NULL-Wert.

INTEGER Der Wert ist eine vorzeichenbehaftete Ganzzahl.

REAL Der Wert ist eine Gleitpunktzahl.

TEXT Der Wert ist ein Textstring (begrenzt auf 256 MB).

BLOB Der Wert ist ein BLOB-Objekt (Binary Large Object, binäres großes Objekt), es handelt sich also um unformatierte Binärdaten (begrenzt auf 256 MB).

Alle Werte, die der Datenbank als Literale bereitgestellt werden, die in eine SQ-Anweisung eingebettet sind, oder Werte, die mithilfe von Parametern an eine vorbereitete SQL-Anweisung gebunden sind, werden einer Speicherklasse zugewiesen, bevor die SQL-Anweisung ausgeführt wird.

Literale, die Teil einer SQL-Anweisung sind, werden den folgenden Speicherklassen zugewiesen: TEXT, wenn sie in einfachen oder doppelten Anführungszeichen stehen; INTEGER, wenn das Literal als Teil einer Zahl ohne Anführungszeichen und ohne Dezimaltrennzeichen oder Exponent angegeben ist; REAL, wenn das Literal eine Zahl ohne Anführungszeichen mit einem Dezimaltrennzeichen oder Exponent ist; NULL, wenn der Wert NULL ist. Literale mit der BLOB-Speicherklasse werden mit der Schreibweise X'ABCD' angegeben. Weitere Informationen finden Sie unter „Literalwerte in Ausdrücken“.

Werte, die mit dem assoziativen SQLStatement.parameters-Array als Parameter bereitgestellt werden, werden der Speicherklasse zugewiesen, die der nativen Datentypbindung am nächsten kommt. Zum Beispiel werden int-Werte als INTEGER-Speicherklasse gebunden, Number-Werte bekommen die REAL-Speicherklasse, String-Werte bekommen die TEXT-Speicherklasse und ByteArray-Objekte bekommen die BLOB-Speicherklasse.

Spaltenaffinität

Die *Affinität* einer Spalte ist der empfohlene Typ für Daten, die in dieser Spalte gespeichert werden. Wenn ein Wert in einer Spalte gespeichert wird (über eine INSERT- oder UPDATE-Anweisung), versucht die Laufzeitumgebung, den Wert von seinem Datentyp in die angegebene Affinität zu konvertieren. Wenn zum Beispiel ein Date-Wert (Datumswert; eine ActionScript- oder JavaScript-Date-Instanz) in eine Spalte eingefügt wird, deren Affinität TEXT ist, wird der Date-Wert in die Stringdarstellung konvertiert (äquivalent zum Aufrufen der toString()-Methode des Objekts), bevor er in der Datenbank gespeichert wird. Wenn der Wert nicht in die angegebene Affinität konvertiert werden kann, tritt ein Fehler auf und der Vorgang wird nicht ausgeführt. Wenn ein Wert mit einer SELECT-Anweisung aus der Datenbank abgerufen wird, wird er als Instanz der Klasse, die der Affinität entspricht, zurückgegeben, unabhängig davon, ob er beim Speichern aus einem anderen Datentyp konvertiert wurde.

Wenn eine Spalte NULL-Werte akzeptiert, kann der ActionScript- oder JavaScript-Wert null als Parameterwert verwendet werden, um NULL in der Spalte zu speichern. Wenn in einer SELECT-Anweisung ein Speicherklassenwert NULL abgerufen wird, wird er immer als der ActionScript- oder JavaScript-Wert null zurückgegeben, unabhängig von der Affinität der Spalte. Wenn eine Spalte NULL-Werte akzeptiert, überprüfen Sie immer die Werte, die aus dieser Spalte abgerufen werden, um festzustellen, ob sie null sind, bevor Sie versuchen, die Werte in einen nicht null-fähigen Typ (zum Beispiel Number oder Boolean) umzuformen.

Jeder Spalte in der Datenbank wird eine der folgenden Typaffinitäten zugewiesen:

- TEXT (oder String)
- NUMERIC
- INTEGER (oder int)
- REAL (oder Number)
- Boolean
- Date
- XML
- XMLLIST
- Object
- NONE

TEXT (oder String)

Eine Spalte mit TEXT- oder String-Affinität speichert alle Daten, die die Speicherklassen NULL, TEXT oder BLOB verwenden. Wenn numerische Daten in eine Spalte mit der TEXT-Affinität eingefügt werden, werden sie vor dem Speichern in Textform konvertiert.

NUMERIC

Eine Spalte mit der NUMERIC-Affinität enthält Werte, die die Speicherklassen NULL, REAL oder INTEGER verwenden. Wenn Textdaten in eine NUMERIC-Spalte eingefügt werden, wird versucht, sie in eine Ganzzahl oder reale Zahl zu konvertieren, bevor sie gespeichert werden. Ist die Konvertierung erfolgreich, wird der Wert mit der INTEGER- oder REAL-Speicherklasse gespeichert (der Wert '10.05' wird zum Beispiel vor dem Speichern in die REAL-Speicherklasse konvertiert). Ist die Konvertierung nicht möglich, kommt es zu einem Fehler. Es wird nicht versucht, einen NULL-Wert zu konvertieren. Ein Wert, der aus einer NUMERIC-Spalte abgerufen wird, wird als Instanz des numerischen Typs zurückgegeben, zu dem der Wert am ehesten passt. Anders ausgedrückt, wenn der Wert eine positive Ganzzahl oder 0 ist, wird er als uint-Instanz zurückgegeben. Wenn der Wert eine negative Ganzzahl ist, wird er als int-Instanz zurückgegeben. Hat der Wert eine Gleitpunktkomponente (ist er also keine Ganzzahl), wird er als Number-Instanz zurückgegeben.

INTEGER (oder int)

Eine Spalte, die die INTEGER-Affinität verwendet, verhält sich wie eine Spalte mit NUMERIC-Affinität, mit einer Ausnahme. Wenn der zu speichernde Wert ein realer Wert (zum Beispiel eine Number-Instanz) ohne Gleitkommakomponente ist oder wenn der Wert ein Textwert ist, der in einen realen Wert ohne Gleitkommakomponente konvertiert werden kann, wird er in eine Ganzzahl konvertiert und mit der INTEGER-Speicherklasse gespeichert. Wenn versucht wird, einen realen Wert mit einer Gleitkommakomponente zu speichern, kommt es zu einem Fehler.

REAL (oder Number)

Eine Spalte mit der REAL- oder NUMBER-Affinität verhält sich wie eine Spalte mit der NUMERIC-Affinität, außer dass sie die Gleitkommadarstellung von Ganzzahlwerten erzwingt. Ein Wert in einer REAL-Spalte wird von der Datenbank immer als Number-Instanz zurückgegeben.

Boolean

Eine Spalte mit Boolean-Affinität speichert true- oder false-Werte. Eine Boolean-Spalte akzeptiert einen Wert, der eine ActionScript- oder JavaScript-Boolean-Instanz ist. Wenn der Code versucht, einen String-Wert zu speichern, wird ein String mit einer Länge größer als null als true betrachtet und ein leerer String als false. Wenn der Code versucht, numerische Daten zu speichern, werden alle Nicht-Null-Werte als true und 0 als false gespeichert. Wenn ein boolescher Wert mit einer SELECT-Anweisung abgerufen wird, wird er als Boolean-Instanz zurückgegeben. Nicht-NULL-Werte werden mit der INTEGER-Speicherklasse gespeichert (0 für false und 1 für true) und in Boolean-Objekte konvertiert, wenn Daten abgerufen werden.

Date

Eine Spalte mit Date-Affinität speichert Datums- und Zeitwerte. Eine Date-Spalte nimmt Werte auf, die ActionScript- oder JavaScript-Date-Instanzen sind. Wenn versucht wird, einen Stringwert in einer Date-Spalte zu speichern, versucht die Laufzeitumgebung, ihn in ein julianisches Datum zu konvertieren. Ist die Konvertierung nicht möglich, kommt es zu einem Fehler. Wenn Code versucht, einen Number-, int- oder uint-Wert zu speichern, wird nicht versucht, die Daten zu validieren, und sie werden als gültiger Julianischer Datumswert betrachtet. Ein Date-Wert, der mit einer SELECT-Anweisung abgerufen wird, wird automatisch in eine Date-Instanz konvertiert. Date-Werte werden mit der REAL-Speicherklasse als julianische Datumswerte gespeichert; Sortier- und Vergleichsoperationen funktionieren also erwartungsgemäß.

XML oder XMLList

Eine Spalte mit XML- oder XMLList-Affinität speichert XML-Strukturen. Wenn Code versucht, Daten mit einem SQLStatement-Parameter in einer XML-Spalte zu speichern, versucht die Laufzeitumgebung, den Wert mit der ActionScript-Funktion XML() oder XMLList() zu konvertieren und zu validieren. Wenn der Wert nicht in gültigen XML-Code konvertiert werden kann, kommt es zu einem Fehler. Wenn beim Versuch, die Daten zu speichern, ein SQL-Literaltextwert verwendet wird (zum Beispiel INSERT INTO (col1) VALUES ('Invalid XML (no closing tag)'), wird der Wert weder analysiert noch validiert; stattdessen wird davon ausgegangen, dass der Wert ein gültiges Format aufweist. Wenn ein ungültiger Wert gespeichert wird, wird er beim Abrufen als leeres XML-Objekt zurückgegeben. XML- und XMLList-Daten werden mit der TEXT-Speicherklasse oder der NULL-Speicherklasse gespeichert.

Object

Eine Spalte mit Object-Affinität speichert komplexe ActionScript- oder JavaScript-Objekte, darunter Object-Klasseninstanzen sowie Instanzen von Object-Unterklassen wie Array-Instanzen und sogar Instanzen benutzerdefinierter Klassen. Object-Spaltendaten werden im AMF3-Format serialisiert und mit der BLOB-Speicherklasse gespeichert. Wenn ein Wert abgerufen wird, wird er aus dem AMF3-Format deserialisiert und als Instanz der Klasse, als die er gespeichert wurde, zurückgegeben. Beachten Sie, dass einige ActionScript-Klassen, insbesondere Anzeigeobjekte, nicht als Instanzen ihres ursprünglichen Datentyps deserialisiert werden. Bevor Sie

Instanzen benutzerdefinierter Klassen speichern, müssen Sie einen Alias für die Klasse registrieren, indem Sie die flash.net.registerClassAlias()-Methode verwenden (oder in Flex [RemoteObject]-Metadaten zur Klassendeklaration hinzufügen). Bevor Sie diese Daten abrufen, müssen Sie denselben Alias für die Klasse registrieren. Alle Daten, die sich nicht korrekt deserialisieren lassen, weil die Klasse an sich nicht deserialisiert werden kann oder weil der Klassenalias fehlt bzw. nicht korrekt ist, werden als anonyme Objekte zurückgegeben (als Object-Klasseninstanz), mit Eigenschaften und Werten, die der ursprünglich gespeicherten Instanz entsprechen.

NONE

Eine Spalte mit NONE-Affinität bevorzugt keine bestimmte Speicherklasse. Es wird nicht versucht, Daten vor dem Einfügen zu konvertieren.

Bestimmen der Affinität

Der Affinitätstyp einer Spalte wird durch den in der CREATE TABLE-Anweisung deklarierten Spaltentyp bestimmt. Beim Bestimmen des Typs werden die folgenden Regeln angewendet (Groß- und Kleinschreibung wird nicht beachtet):

- Wenn der Datentyp der Spalte den String "CHAR", "CLOB", "STRI" oder "TEXT" enthält, hat die Spalte die TEXT/String-Affinität. Beachten Sie, dass der Typ VARCHAR den String "CHAR" enthält; deshalb wird ihm die TEXT-Affinität zugewiesen.
- Wenn der Datentyp der Spalte den String "BLOB" enthält oder wenn kein Datentyp angegeben ist, hat die Spalte die NONE-Affinität.
- Wenn der Datentyp der Spalte den String "XMLL" enthält, hat die Spalte die XMLList-Affinität.
- Wenn der Datentyp den String "XML" enthält, hat die Spalte die XML-Affinität.
- Wenn der Datentyp den String "OBJE" enthält, hat die Spalte die Object-Affinität.
- Wenn der Datentyp den String "BOOL" enthält, hat die Spalte die Boolean-Affinität.
- Wenn der Datentyp den String "DATE" enthält, hat die Spalte die Date-Affinität.
- Wenn der Datentyp den String "INT" (oder "UINT") enthält, gilt die INTEGER/int-Affinität.
- Wenn der Datentyp der Spalte den String "REAL", "NUMB", "FLOA" oder "DOUB" enthält, hat die Spalte die REAL/Number-Affinität.
- Andernfalls ist die Affinität NUMERIC.
- Wenn eine Tabelle mit einer CREATE TABLE t AS SELECT...-Anweisung erstellt wird, wird für keine der Spalten ein Datentyp festgelegt und sie erhalten die NONE-Affinität.

Datentypen und Vergleichsoperatoren

Die folgenden binären Vergleichsoperatoren werden unterstützt: =, <, <=, >= und !=. Außerdem besteht Unterstützung für eine Operation zum Testen der set-Mitgliedschaft, für IN und für den tenären Vergleichsoperator BETWEEN. Weitere Informationen zu diesen Operatoren finden Sie unter „Operatoren“.

Die Ergebnisse eines Vergleichs sind von der Speicherklasse der beiden verglichenen Werte abhängig. Wenn zwei Werte verglichen werden, gelten die folgenden Regeln:

- Ein Wert mit der NULL-Speicherklasse wird als kleiner als jeder andere Wert betrachtet (einschließlich anderer Werte mit der NULL-Speicherklasse).
- Ein INTEGER- oder REAL-Wert wird als kleiner als jeder TEXT- oder BLOB-Wert betrachtet. Wenn ein INTEGER- oder REAL-Wert mit einem anderen INTEGER- oder REAL-Wert verglichen wird, wird ein numerischer Vergleich ausgeführt.

- Ein TEXT-Wert ist kleiner als ein BLOB-Wert. Wenn zwei TEXT-Werte verglichen werden, wird ein binärer Vergleich ausgeführt.
- Wenn zwei BLOB-Werte verglichen werden, wird das Ergebnis immer mithilfe eines binären Vergleichs ermittelt.

Der ternäre BETWEEN-Operator wird immer in den äquivalenten binären Ausdruck umgeformt. Beispielsweise wird „a BETWEEN b AND c“ in „a >= b AND a <= c“ umgewandelt, selbst wenn dies bedeutet, dass bei jedem der Vergleiche, die zum Auswerten des Ausdrucks erforderlich sind, eine andere Affinität auf „a“ angewendet wird.

Ausdrücke des Typs „a IN (SELECT b ....)“ werden gemäß den drei Regeln gehandhabt, die bereits für Binärvergleiche genannt wurden, also ähnlich wie „a = b“. Beispiel: Wenn „b“ ein Spaltenwert und „a“ ein Ausdruck ist, wird die Affinität von „b“ auf „a“ angewendet, bevor Vergleiche durchgeführt werden. Der Ausdruck „a IN (x, y, z)“ wird in die Form „a = +x OR a = +y OR a = +z“ umgewandelt. Die Werte rechts vom IN-Operator (in diesem Beispiel x, y und z) gelten als Ausdrücke, selbst wenn sie eigentlich Spaltenwerte sind. Wenn der Wert links des IN-Operators eine Spalte ist, wird die Affinität dieser Spalte verwendet. Wenn der Wert ein Ausdruck ist, findet keine Konvertierung statt.

Wie Vergleiche ausgeführt werden, kann auch von der Verwendung einer COLLATE-Klausel beeinflusst werden. Weitere Informationen finden Sie unter COLLATE.

Datentypen und mathematische Operatoren

Für jeden der unterstützten mathematischen Operatoren *, /, %, + und - wird die numerische Affinität auf jeden Operanden angewendet, bevor der Ausdruck evaluiert wird. Wenn ein Operand nicht erfolgreich in die NUMERIC-Speicherklasse konvertiert werden kann, wird der Ausdruck als NULL evaluiert.

Wenn der Verkettungsoperator || verwendet wird, wird jeder Operand in die TEXT-Speicherklasse konvertiert, bevor der Ausdruck evaluiert wird. Wenn ein Operand nicht in die TEXT-Speicherklasse konvertiert werden kann, ist das Ergebnis des Ausdrucks NULL. Es kann in zwei Situationen vorkommen, dass der Wert nicht konvertiert werden kann: wenn der Wert des Operanden NULL ist oder wenn es sich um ein BLOB handelt, das eine Nicht-TEXT-Speicherklasse enthält.

Datentypen und Sortierung

Wenn Werte durch eine ORDER BY-Klausel sortiert werden, stehen die Werte mit der NULL-Speicherklasse an erster Stelle. Danach folgen die INTEGER- und REAL-Werte in numerischer Reihenfolge, gefolgt von TEXT-Werten in binärer Reihenfolge oder nach der angegebenen Überprüfung (BINARY oder NOCASE) geordnet. Zum Schluss folgen die BLOB-Werte in binärer Reihenfolge. Vor der Sortierung werden keinen Klassenkonvertierungen ausgeführt.

Datentypen und Gruppierung

Wenn Werte mit der GROUP BY-Klausel gruppiert werden, werden Werte mit unterschiedlichen Speicherklassen als verschieden betrachtet. Eine Ausnahme sind INTEGER- und REAL-Werte, die als gleich betrachtet werden, wenn sie numerische Äquivalente sind. Auf Werte, die Ergebnis der GROUP BY-Klausel sind, werden keine Affinitäten angewendet.

Datentypen und zusammengesetzte SELECT-Anweisungen

Die zusammengesetzten SELECT-Operatoren UNION, INTERSECT und EXCEPT führen implizite Vergleiche zwischen Werten aus. Bevor diese Vergleiche ausgeführt werden, können auf die einzelnen Werte Affinitäten angewendet werden. Es wird dieselbe Affinität auf alle Werte angewendet, die in einer einzelnen Spalte des Ergebnissatzes des zusammengesetzten SELECT-Operators zurückgegeben werden. Die angewendete Affinität ist die Affinität der Spalte, die von der ersten Komponenten-SELECT-Anweisung zurückgegeben wird, die einen Spaltenwert an dieser Position hat (und keinen anderen Ausdruck). Wenn für eine bestimmte zusammengesetzte SELECT-Spalte keine der Komponenten-SELECT-Anweisungen einen Spaltenwert zurückgibt, wird keine Affinität auf die Werte dieser Spalte angewendet, bevor sie verglichen werden.

Kapitel 67: SQL-Fehlerdetailmeldungen, IDs und Argumente

Die SQLError-Klasse repräsentiert verschiedene Fehler, die bei der Arbeit mit einer lokalen SQL-Datenbank in Adobe AIR auftreten können. Für einen beliebigen gegebenen Ausdruck verfügt die SQLError-Instanz über eine `details`-Eigenschaft mit einer Fehlermeldung. Zusätzlich hat jede Fehlermeldung einen eindeutigen Bezeichner, der in der `detailID`-Eigenschaft des SQLError-Objekts zur Verfügung steht. Mithilfe der `detailID`-Eigenschaft kann eine Anwendung die spezifische details-Fehlermeldung identifizieren. Die Anwendung kann alternativen Text für den Endbenutzer in der Sprache seines jeweiligen Gebietsschemas bereitstellen. Die Argumentwerte im `detailArguments`-Array können an der richtigen Position im Fehlermeldungsstring ersetzt werden. Dies ist hilfreich bei Anwendungen, bei denen die Fehlermeldung der `details`-Eigenschaft direkt für Endbenutzer in einem bestimmten Gebietsschema angezeigt wird.

In der folgenden Tabelle sind die `detailID`-Werte und der Text der dazugehörigen Fehlermeldungen aufgeführt. Platzhalter in den Meldungen geben an, wo zur Laufzeit `detailArguments`-Werte eingesetzt werden. Diese Liste können Sie als Quelle für die Lokalisierung der Fehlermeldungen verwenden, die in SQL-Datenbankvorgängen auftreten können.

SQLError – detailID	Ausführliche Fehlermeldung und Parameter
1001	Connection closed. (Verbindung geschlossen.)
1102	Database must be open to perform this operation. (Die Datenbank muss geöffnet sein, um diesen Vorgang auszuführen.)
1003	%s [,|and %s] parameter name(s) found in parameters property but not in the SQL specified. (Parametername(n) %s [,|und %s] in parameters-Eigenschaft gefunden, aber nicht in der angegebenen SQL.)
1004	Mismatch in parameter count. (Abweichung in Parameterzählung.) Found %d in SQL specified and %d value(s) set in parameters property. Expecting values for %s [,|and %s]. (%d in angegebener SQL gefunden und %d Wert(e) in der parameters-Eigenschaft festgelegt. Erwartet werden Werte für %s [,|und %s].)
1005	Auto compact could not be turned on. (Auto-Kompakt konnte nicht eingeschaltet werden.)
1006	The pageSize value could not be set. (Der pageSize-Wert konnte nicht festgelegt werden.)
1007	The schema object with name '%s' of type '%s' in database '%s' was not found. (Das Schemaobjekt mit dem Namen '%s' des Typs '%s' in Datenbank '%s' wurde nicht gefunden.)
1008	The schema object with name '%s' in database '%s' was not found. (Das Schemaobjekt mit dem Namen '%s' in Datenbank '%s' wurde nicht gefunden.)
1009	No schema objects with type '%s' in database '%s' were found. (Es wurden keine Schemaobjekte des Typs '%s' in Datenbank '%s' gefunden.)
1010	No schema objects in database '%s' were found. (Es wurden keine Schemaobjekte in Datenbank '%s' gefunden.)
2001	Parser stack overflow. (Parser-Stapelüberlauf.)
2002	Too many arguments on function '%s'. (Zu viele Argumente für Funktion '%s').
2003	near '%s': syntax error. (In der Nähe von '%s': Syntaxfehler.)
2004	there is already another table or index with this name: '%s' (Es ist bereits eine andere Tabelle oder ein anderer Index mit diesem Namen vorhanden: '%s'.)
2005	PRAGMA is not allowed in SQL. (PRAGMA ist in SQL nicht zulässig.)
2006	Not a writable directory. (Schreiben in das Verzeichnis ist nicht möglich.)
2007	Unknown or unsupported join type: '%s %s %s'. (Unbekannter oder nicht unterstützter Verbindungstyp: '%s %s %s'.)
2008	RIGHT and FULL OUTER JOINs are not currently supported. (RIGHT und FULL OUTER JOINs werden zurzeit nicht unterstützt.)

2009	A NATURAL join may not have an ON or USING clause. (Eine NATURAL-Verbindung darf keine ON- oder USING-Klausel enthalten.)
2010	Cannot have both ON and USING clauses in the same join. (Eine Verbindung kann nicht sowohl eine ON-Klausel als auch eine USING-Klausel enthalten.)
2011	Cannot join using column '%s' - column not present in both tables. (Verbinden mit Spalte '%s' nicht möglich – Spalte ist nicht in beiden Tabellen vorhanden.)
2012	Only a single result allowed for a SELECT that is part of an expression. (Für eine SELECT-Anweisung, die zu einem Ausdruck gehört, ist nur ein einzelnes Ergebnis zulässig.)
2013	No such table: '[%s.]%s' (Keine derartige Tabelle vorhanden: '[%s.]%s'.)
2014	No tables specified. (Keine Tabellen angegeben.)
2015	Too many columns in result set|too many columns on '%s'. (Zu viele Spalten in Ergebnismenge|zu viele Spalten in '%s'.)
2016	%s ORDER|GROUP BY term out of range - should be between 1 and %d (ORDER|GROUP BY-Begriff %s außerhalb des gültigen Bereichs – sollte zwischen 1 und %d liegen.)
2017	Too many terms in ORDER BY clause. (Zu viele Begriffe in ORDER BY-Klausel.)
2018	%s ORDER BY term out of range - should be between 1 and %d. (%s ORDER BY-Begriff außerhalb des zulässigen Bereichs – muss zwischen 1 und %d liegen.)
2019	%r ORDER BY term does not match any column in the result set. (%r ORDER BY-Begriff entspricht keiner Spalte in der Ergebnismenge.)
2020	ORDER BY clause should come after '%s' not before. (ORDER BY-Klausel sollte hinter '%s' stehen, nicht davor.)
2021	LIMIT clause should come after '%s' not before. (LIMIT-Klausel sollte hinter '%s' stehen, nicht davor.)
2022	SELECTs to the left and right of '%s' do not have the same number of result columns. (SELECTs links und rechts von '%s' haben nicht dieselbe Anzahl Ergebnisspalten.)
2023	A GROUP BY clause is required before HAVING. (Eine GROUP BY-Klausel ist vor HAVING erforderlich.)
2024	Aggregate functions are not allowed in the GROUP BY clause. (Aggregatfunktionen sind in der GROUP BY-Klausel nicht zulässig.)
2025	DISTINCT in aggregate must be followed by an expression. (Auf DISTINCT in Aggregation muss ein Ausdruck folgen.)
2026	Too many terms in compound SELECT. (Zu viele Begriffe in zusammengesetzter SELECT-Anweisung.)
2027	Too many terms in ORDER|GROUP BY clause. (Zu viele Begriffe in ORDER|GROUP BY-Klausel.)
2028	Temporary trigger may not have qualified name. (Temporärer Auslöser darf keinen qualifizierten Namen haben.)
2030	Trigger '%s' already exists. (Auslöser '%s' existiert bereits.)
2032	Cannot create BEFORE|AFTER trigger on view: '%s'. (BEFORE|AFTER-Auslöser kann nicht erstellt werden für Ansicht: '%s'.)
2033	Cannot create INSTEAD OF trigger on table: '%s'. (INSTEAD OF-Auslöser kann nicht erstellt werden für Tabelle: '%s'.)
2034	No such trigger: '%s'. (Kein derartiger Auslöser vorhanden: '%s'.)
2035	Recursive triggers not supported ('%s'). (Rekursive Auslöser werden nicht unterstützt ('%s').)
2036	No such column: (Keine derartige Spalte vorhanden:) %s[.%s[.%s]]
2037	VACUUM is not allowed from SQL. (VACUUM ist von SQL nicht zulässig.)
2043	Table '%s': indexing function returned an invalid plan. (Tabelle '%s': Indexfunktion hat einen ungültigen Plan zurückgegeben.)
2044	At most %d tables in a join. (Höchstens %d Tabellen in einer Verbindung.)
2046	Cannot add a PRIMARY KEY column. (PRIMARY KEY-Spalte kann nicht hinzugefügt werden.)
2047	Cannot add a UNIQUE column. (UNIQUE-Spalte kann nicht hinzugefügt werden.)
2048	Cannot add a NOT NULL column with default value NULL. (Es kann keine NOT NULL-Spalte mit Standardwert NULL hinzugefügt werden.)
2049	Cannot add a column with non-constant default. (Es kann keine Spalte mit nicht konstantem Standard hinzugefügt werden.)
2050	Cannot add a column to a view. (Es kann keine Spalte zur Ansicht hinzugefügt werden.)
2051	ANALYZE is not allowed in SQL. (ANALYZE ist in SQL nicht zulässig.)

2052	Invalid name: '%s'. (Ungültiger Name: '%s'.)
2053	ATTACH is not allowed from SQL. (ATTACH ist von SQL nicht zulässig.)
2054	%s '%s' cannot reference objects in database '%s' (%s '%s' kann nicht auf Objekte in Datenbank '%s' verweisen.)
2055	Access to '[%s.]%s.%s' is prohibited. (Zugriff auf '[%s.]%s.%s' ist untersagt.)
2056	Not authorized. (Nicht autorisiert.)
2058	No such view: '[%s.]%s'. (Keine derartige Ansicht vorhanden: '[%s.]%s'.)
2060	Temporary table name must be unqualified. (Der Name einer temporären Tabelle darf nicht qualifiziert sein.)
2061	Table '%s' already exists. (Tabelle '%s' existiert bereits.)
2062	There is already an index named: '%s'. (Ein Index mit folgendem Namen ist bereits vorhanden: '%s'.)
2064	Duplicate column name: '%s'. (Doppelter Spaltenname: '%s'.)
2065	Table '%s' has more than one primary key. (Tabelle '%s' hat mehr als einen Primärschlüssel.)
2066	AUTOINCREMENT is only allowed on an INTEGER PRIMARY KEY (AUTOINCREMENT ist nur für einen INTEGER PRIMARY KEY zulässig.)
2067	No such collation sequence: '%s'. (Keine derartige Vergleichsreihenfolge vorhanden: '%s'.)
2068	Parameters are not allowed in views. (Parameter sind in Ansichten nicht zulässig.)
2069	View '%s' is circularly defined. (Ansicht '%s' ist zirkulär definiert.)
2070	Table '%s' may not be dropped. (Tabelle '%s' darf nicht entfernt werden.)
2071	Use DROP VIEW to delete view '%s'. (DROP VIEW zum Löschen von Ansicht '%s' verwenden.)
2072	Use DROP TABLE to delete table '%s'. (DROP TABLE zum Löschen von Tabelle '%s' verwenden.)
2073	Foreign key on '%s' should reference only one column of table '%s'. (Fremdschlüssel für '%s' darf nur auf eine Spalte der Tabelle '%s' verweisen.)
2074	Number of columns in foreign key does not match the number of columns in the referenced table. (Anzahl der Spalten im Fremdschlüssel stimmt nicht mit der Anzahl der Spalten in der referenzierten Tabelle überein.)
2075	Unknown column '%s' in foreign key definition. (Unbekannte Spalte '%s' in Fremdschlüsseldefinition.)
2076	Table '%s' may not be indexed. (Tabelle '%s' darf nicht indiziert werden.)
2077	Views may not be indexed. (Ansichten dürfen nicht indiziert werden.)
2080	Conflicting ON CONFLICT clauses specified. (Widersprüchliche ON CONFLICT-Klauseln angegeben.)
2081	No such index: '%s'. (Kein derartiger Index vorhanden: '%s'.)
2082	Index associated with UNIQUE or PRIMARY KEY constraint cannot be dropped. (Der mit der UNIQUE- oder PRIMARY KEY-Beschränkung verknüpfte Index kann nicht entfernt werden.)
2083	BEGIN is not allowed in SQL. (BEGIN ist in SQL nicht zulässig.)
2084	COMMIT is not allowed in SQL. (COMMIT ist in SQL nicht zulässig.)
2085	ROLLBACK is not allowed in SQL. (ROLLBACK ist in SQL nicht zulässig.)
2086	Unable to open a temporary database file for storing temporary tables. (Temporäre Datenbankdatei kann nicht zum Speichern von temporären Tabellen geöffnet werden.)
2087	Unable to identify the object to be reindexed. (Das neu zu indizierende Objekt kann nicht identifiziert werden.)
2088	Table '%s' may not be modified. (Tabelle '%s' darf nicht geändert werden.)
2089	Cannot modify '%s' because it is a view. ('%s' kann nicht geändert werden, da es sich dabei um eine Ansicht handelt.)
2090	Variable number must be between ?0 and ?%d<. (Variablennummer muss zwischen ?0 und ?%d< liegen.)
2092	Misuse of aliased aggregate '%s'. (Falsche Verwendung der Alias-Aggregatfunktion '%s'.)
2093	Ambiguous column name: '[%s.[%s.]]%s'. (Nicht eindeutiger Spaltenname '[%s.[%s.]]%s'.)
2094	No such function: '%s'. (Keine derartige Funktion vorhanden: '%s'.)
2095	Wrong number of arguments to function '%s'. (Falsche Anzahl an Argumenten für Funktion '%s'.)

2096	Subqueries prohibited in CHECK constraints. (Unterabfragen sind in CHECK-Beschränkungen nicht zulässig.)
2097	Parameters prohibited in CHECK constraints. (Parameter sind in CHECK-Beschränkungen nicht zulässig.)
2098	Expression tree is too large (maximum depth %d) (Ausdruckstruktur ist zu groß (maximale Tiefe %d))
2099	RAISE() may only be used within a trigger-program (RAISE() kann nur innerhalb eines Auslöserprogramms verwendet werden.)
2100	Table '%s' has %d columns but %d values were supplied. (Tabelle '%s' hat %d Spalten, aber %d Werte wurden angegeben.)
2101	Database schema is locked: '%s'. (Datenbankschema ist gesperrt: '%s'.)
2102	Statement too long. (Anweisung ist zu lang.)
2103	Unable to delete/modify collation sequence due to active statements (Aufgrund aktiver Anweisungen kann die Überprüfungssequenz nicht gelöscht/geändert werden.)
2104	Too many attached databases - max %d. (Zu viele angefügte Datenbanken – max %d.)
2105	Cannot ATTACH database within transaction. (ATTACH kann für Datenbank innerhalb der Transaktion nicht ausgeführt werden.)
2106	Database '%s' is already in use. (Datenbank '%s' wird bereits verwendet.)
2108	Attached databases must use the same text encoding as main database. (Die angefügte Datenbank muss dieselbe Textkodierung verwenden wie die Hauptdatenbank.)
2200	Out of memory. (Zu wenig Speicher.)
2201	Unable to open database. (Datenbank kann nicht geöffnet werden.)
2202	Cannot DETACH database within transaction. (DETACH kann für Datenbank innerhalb der Transaktion nicht ausgeführt werden.)
2203	Cannot detach database: '%s'. (Datenbank kann nicht getrennt werden: '%s'.)
2204	Database '%s' is locked. (Datenbank '%s' ist gesperrt.)
2205	Unable to acquire a read lock on the database. (Es kann keine Lesesperre für die Datenbank verwendet werden.)
2206	[column|columns] '%s'[,'%s'] are not [unique|is] not unique. ([Spalte|Spalten] '%s'[,'%s'] sind nicht [eindeutig|ist] nicht eindeutig.)
2207	Malformed database schema. (Datenbankschema hat ein ungültiges Format.)
2208	Unsupported file format. (Dateiformat wird nicht unterstützt.)
2209	Unrecognized token: '%s'. (Nicht erkanntes Token: '%s'.)
2300	Could not convert text value to numeric value. (Textwert konnte nicht in numerischen Wert konvertiert werden.)
2301	Could not convert string value to date. (Stringwert konnte nicht in Datum konvertiert werden.)
2302	Could not convert floating point value to integer without loss of data. (Gleitpunktwert kann nicht ohne Datenverlust in eine Ganzzahl konvertiert werden.)
2303	Cannot rollback transaction - SQL statements in progress. (Transaktion kann nicht rückgängig gemacht werden – SQL-Anweisung wird gerade ausgeführt.)
2304	Cannot commit transaction - SQL statements in progress. (Transaktion kann nicht übernommen werden – SQL-Anweisung wird gerade ausgeführt.)
2305	Database table is locked: '%s'. (Datenbanktabelle ist gesperrt: '%s'.)
2306	Read-only table. (Schreibgeschützte Tabelle.)
2307	String or blob too big. (String oder Blob zu groß.)
2309	Cannot open indexed column for writing. (Indizierte Spalte kann nicht zum Schreiben geöffnet werden.)
2400	Cannot open value of type %s. (Wert des Typs %s kann nicht geöffnet werden.)
2401	No such rowid: %s<. (Keine derartige Zeilen-ID vorhanden: %s<.)
2402	Object name reserved for internal use: '%s'. (Objektname für interne Verwendung reserviert: '%s'.)
2403	View '%s' may not be altered. (Ansicht '%s' darf nicht geändert werden.)
2404	Default value of column '%s' is not constant. (Standardwert der Spalte '%s' ist nicht konstant.)
2405	Not authorized to use function '%s'. (Nicht zur Verwendung der Funktion '%s' autorisiert.)
2406	Misuse of aggregate function '%s'. (Falsche Verwendung der Aggregatfunktion '%s'.)

2407	Misuse of aggregate: '%s'. (Falsche Verwendung von Aggregat: '%s'.)
2408	No such database: '%s'. (Keine derartige Datenbank vorhanden: '%s'.)
2409	Table '%s' has no column named '%s'. (Tabelle '%s' hat keine Spalte namens '%s'.)
2501	No such module: '%s'. (Kein derartiges Modul vorhanden: '%s'.)
2508	No such savepoint: '%s'. (Kein derartiger Speicherpunkt vorhanden: '%s'.)
2510	Cannot rollback - no transaction is active. (Rückgängigmachen nicht möglich – keine Transaktion aktiv.)
2511	Cannot commit - no transaction is active. (Übernehmen nicht möglich – keine Transaktion aktiv.)

Kapitel 68: Adobe Graphics Assembly Language (AGAL)

Die Adobe Graphics Assembly Language (AGAL) ist eine Shader-Sprache zum Definieren von Vertex- und Fragment-Renderingprogrammen. Die AGAL-Programme müssen im hier beschriebenen binären Bytecodeformat in den Renderingkontext hochgeladen werden.

AGAL-Bytecodeformat

AGAL-Bytecode muss das Format Endian.LITTLE_ENDIAN verwenden.

Bytecode-Header

AGAL-Bytecode muss mit einem 7-Byte-Header beginnen:

```
A0 01000000 A1 00 -- for a vertex program
A0 01000000 A1 01 -- for a fragment program
```

Offset (Bytes)	Größe (Bytes)	Name	Beschreibung
0	1	magic	muss 0xa0 sein
1	4	version	muss 1 sein
5	1	Shadertyp-ID	muss 0xa1 sein
6	1	Shadertyp	0 für ein Vertexprogramm; 1 für ein Fragmentprogramm

Token

Dem Header folgt sofort eine beliebige Anzahl Token. Jedes Token ist 192 Bit (24 Byte) groß und hat immer folgendes Format:

```
[opcode][destination][source1][source2 oder sampler]
```

Nicht jeder Opcode verwendet alle diese Felder. Nicht verwendete Felder müssen auf 0 gesetzt werden.

Opcodes

Das [opcode]-Feld ist 32 Bit groß und kann einen dieser Werte aufweisen:

Name	Opcode	Methode	Beschreibung
mov	0x00	verschieben	Daten von source1 nach destination verschieben, komponentenweise
add	0x01	addieren	destination = source1 + source2, komponentenweise
sub	0x02	subtrahieren	destination = source1 - source2, komponentenweise
mul	0x03	multiplizieren	destination = source1 * source2, komponentenweise
div	0x04	dividieren	destination = source1 / source2, komponentenweise
rcp	0x05	Kehrwert	destination = 1/source1, komponentenweise

Name	Opcode	Methode	Beschreibung
min	0x06	minimum	destination = minimum(source1,source2), komponentenweise
max	0x07	maximum	destination = maximum(source1,source2), komponentenweise
frc	0x08	fraktional	destination = source1 - (float)floor(source1), komponentenweise
sqt	0x09	Quadratwurzel	destination = sqrt(source1), komponentenweise
rsq	0x0a	Quadratwurzel-Kehrwert	destination = 1/sqrt(source1), komponentenweise
pow	0x0b	Potenz	destination = pow(source1,source2), komponentenweise
log	0x0c	Logarithmus	destination = log_2(source1), komponentenweise
exp	0x0d	Exponential	destination = 2^source1, komponentenweise
nrm	0x0e	normalisieren	destination = normalize(source1), komponentenweise (erzeugt nur ein 3-Komponenten-Ergebnis, das Ziel muss auf .xyz oder weniger maskiert werden)
sin	0x0f	Sinus	destination = sin(source1), komponentenweise
cos	0x10	Kosinus	destination = cos(source1), komponentenweise
crs	0x11	Kreuzprodukt	destination.x = source1.y * source2.z - source1.z * source2.y destination.y = source1.z * source2.x - source1.x * source2.z destination.z = source1.x * source2.y - source1.y * source2.x (erzeugt nur ein 3-Komponenten-Ergebnis, das Ziel muss auf .xyz oder weniger maskiert werden)
dp3	0x12	Skalarprodukt	destination = source1.x*source2.x + source1.y*source2.y + source1.z*source2.z
dp4	0x13	Skalarprodukt	destination = source1.x*source2.x + source1.y*source2.y + source1.z*source2.z + source1.w*source2.w
abs	0x14	Absolut	destination = abs(source1), komponentenweise
neg	0x15	Gegenzahl	destination = -source1, komponentenweise
sat	0x16	sättigen	destination = maximum(minimum(source1,1),0), komponentenweise
m33	0x17	Matrix multiplizieren 3x3	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) (erzeugt nur ein 3-Komponenten-Ergebnis, das Ziel muss auf .xyz oder weniger maskiert werden)
m44	0x18	Matrix multiplizieren 4x4	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) + (source1.w * source2[0].w) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) + (source1.w * source2[1].w) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) + (source1.w * source2[2].w) destination.w = (source1.x * source2[3].x) + (source1.y * source2[3].y) + (source1.z * source2[3].z) + (source1.w * source2[3].w)

Name	Opcode	Methode	Beschreibung
m34	0x19	Matrix multiplizieren 3x4	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) + (source1.w * source2[0].w) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) + (source1.w * source2[1].w) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) + (source1.w * source2[2].w) (erzeugt nur ein 3-Komponenten-Ergebnis, das Ziel muss auf .xyz oder weniger maskiert werden)
kil	0x27	verwerfen (nur Fragment-Shader)	Wenn eine Einzelskalar-Ausgabekomponente kleiner als null ist, wird das Fragment verworfen und nicht in den Framebuffer gezeichnet. (Zielregister muss vollständig auf 0 gesetzt werden)
tex	0x28	Texturbeispiel (nur Fragment-Shader)	destination entspricht dem Laden aus Textur source2 bei den Koordinaten source1. In diesem Fall muss source2 im Samplerformat vorliegen.
sge	0x29	festlegen, falls größer oder gleich	destination = source1 >= source2 ? 1 : 0, komponentenweise
slt	0x2a	festlegen, falls kleiner als	destination = source1 < source2 ? 1 : 0, komponentenweise
seq	0x2c	festlegen, falls gleich	destination = source1 == source2 ? 1 : 0, komponentenweise
sne	0x2d	festlegen, falls nicht gleich	destination = source1 != source2 ? 1 : 0, komponentenweise

Zielfeldformat

Das [destination]-Feld hat eine Größe von 32 Bit:

```
31.............................0
----TTTT----MMMMNNNNNNNNNNNNNNNN
```

T = Registertyp (4 Bits)

M = Schreibmaske (4 Bits)

N = Registernummer (16 Bits)

- = nicht definiert, muss 0 sein

Quellenfeldformat

Das [source]-Feld hat eine Größe von 64 Bit:

```
63............................................................0
D------------QQ----IIII----TTTTSSSSSSSSOOOOOOOONNNNNNNNNNNNNNNN
```

D = Direkt=0/Indirekt=1 für direkt Q und I werden ignoriert, 1Bit

Q = Indexregisterkomponentenauswahl (2 Bits)

I = Indexregistertype (4 Bits)

T = Registertyp (4 Bits)

S = Swizzle (8 Bits, 2 Bits pro Komponente)

O = Indirekter Offset (8 Bits)

N = Registernummer (16 Bits)

- = nicht definiert, muss 0 sein

Samplerfeldformat

Das zweite Quellfeld für den tex-Opcode muss im [sampler]-Format vorliegen, welches 64 Bit groß ist:

```
63.............................................................0
FFFFMMMMWWWWSSSSDDDD--------TTTT--------BBBBBBBBNNNNNNNNNNNNNNNN
```

N = Samplerregisternummer (16 Bits)

B = Textur Level-of-detail (LOD) Bias, vorzeichenbehaftete Ganzzahl, skalieren um 8. Der verwendete Gleitkommawert ist b/8.0 (8 Bits)

T = Registertyp, muss 5 sein, Sampler (4 Bits)

F = Filter (0=nächste,1=linear) (4 Bits)

M = Mipmap (0=deaktivieren,1=nächste, 2=linear)

W = Wrapping (0=fixieren,1=wiederholen)

S = Spezielle Flag-Bits (muss 0 sein)

D = Dimension (0=2D, 1=Würfel)

Programmregister

Die folgenden Registertypen sind definiert. Verwenden Sie den aufgelisteten Wert, um einen Registertyp in den Feldern eines Tokens anzugeben:

Name	Wert	Anzahl pro Fragmentprogramm	Anzahl pro Vertexprogramm	Verwendung
Attribute	0	–	8	Vertexshadereingabe; aus einem Vertexbuffer lesen, der mit Context3D.setVertexBufferAt() angegeben wird.
Constant	1	28	128	Shadereingabe; mit der Context3D.setProgramConstants()-Funktionsfamilie festlegen.
Temporary	2	8	8	Temporäres Register für die Berechnung, außerhalb des Programms nicht zugänglich.
Output	3	1	1	Shaderausgabe: in einem Vertexprogramm ist die Ausgabe die Clipspaceposition; in einem Fragmentprogramm ist die Ausgabe eine Farbe.
Varying	4	8	8	Interpolierte Daten zwischen Vertex- und Fragmentprogrammen übertragen. Die unterschiedlichen Register aus dem Vertexprogramm werden als Eingabe in das Fragmentprogramm angewendet. Die Werte werden entsprechend dem Abstand von den Dreiecksscheitelpunkten interpoliert.
Sampler	5	8	–	Fragmentshadereingabe; aus einer mit Context3D.setTextureAt() angegebenen Textur lesen.