Guia do desenvolvedor do ACTIONSCRIPT® 3.0

Avisos legais

Para ver os avisos legais, consulte http://help.adobe.com/pt_BR/legalnotices/index.html.

Última atualização em 14/10/2015

Conteúdo

Capítulo 1: Trabalho com datas e horas
Gerenciamento de datas de calendário e horas 1
Controle de intervalos de tempo 4
Exemplo de data e hora: relógio analógico simples 6

Capítulo 2: Trabalho com sequências de caracteres
Noções básicas de strings 10
Criação de strings 11
A propriedade length 12
Trabalho com caracteres em strings 12
Comparação de strings 13
Obtenção de representações de strings de outros objetos 14
Concatenação de strings 14
Localização de substrings e padrões em strings 15
Conversão de strings entre maiúsculas e minúsculas 19
Exemplos de sequências de caracteres: arte ASCII 19

Capítulo 3: Trabalho com matrizes
Noções básicas sobre matrizes 25
Matrizes indexadas 27
Matrizes associativas 38
Matrizes multidimensionais 41
Clonagem de matrizes 43
Extensão da classe Array 43
Exemplo de matriz: PlayList 48

Capítulo 4: Manipulação de erros
Noções básicas da manipulação de erros 52
Tipos de erros 53
Manipulação de erros no ActionScript 3.0 56
Trabalho com as versões de depurador dos tempos de execução do Flash 57
Manipulação de erros síncronos em um aplicativo 58
Criação de classes de erros personalizadas 62
Resposta a eventos e status de erros 63
Comparação das classes Error 67
Exemplo de tratamento de erros: aplicativo CustomErrors 70

Capítulo 5: Uso de expressões regulares
Noções básicas de expressões regulares: 76
Sintaxe da expressão regular 77
Métodos para usar expressões regulares com strings 90
Exemplo de expressões regulares: um analisador Wiki 92

Capítulo 6: Trabalho com XML
Noções básicas sobre XML 96
A abordagem E4X em relação ao processamento de XML 99
Objetos XML 100
Objetos XMLList 103
Inicialização de variáveis XML 104
Montagem e transformação de objetos XML 105
Como percorrer estruturas XML 107
Uso de namespaces XML 111
Conversão de tipo XML 112
Leitura de documentos XML externos 113
Exemplo XML no ActionScript: carregamento de dados RSS da Web 114

Capítulo 7: Uso da funcionalidade JSON nativa
Visão geral da API JSON 118
Definição do comportamento personalizado de JSON 119

Capítulo 8: Manipulação de eventos
Noções básicas sobre a manipulação de eventos 126
Como a manipulação de eventos do ActionScript 3.0 difere das versões anteriores 128
O fluxo de eventos 130
Objetos de evento 132
Ouvintes de evento 136
Exemplo de tratamento de evento: relógio de alarme 142

Capítulo 9: Trabalhar com domínios de aplicativo

Capítulo 10: Programação de exibição
Noções básicas sobre a programação de exibição 153
Principais classes de exibição 156
Vantagens da abordagem da lista de exibição 158
Trabalho com os objetos de exibição 160
Manipulação de objetos de exibição 175
Animação de objetos 196
Orientação do palco 199
Carregamento dinâmico do conteúdo da exibição 202
Exemplo de objeto de exibição: SpriteArranger 207

Capítulo 11: Trabalho com geometria
Noções básicas de geometria 214
Uso de objetos Point 215
Uso de objetos Rectangle 217
Uso de objetos Matrix 220
Exemplo de geometria: aplicação de uma transformação de matriz em um objeto de exibição 221

Capítulo 12: Uso da API de desenho
Noções básicas da API de desenho 226
A classe Graphics 227

Desenho de linhas e curvas 227
Desenho de formas utilizando os métodos incorporados 230
Criação de linhas e preenchimentos gradientes 231
Uso da classe Math com métodos de desenho 235
Animação com a API de desenho 236
Exemplo de API de desenho: gerador visual algorítmico 237
Uso avançado da API de desenho 239

Capítulo 13: Trabalho com bitmaps
Noções básicas do trabalho com bitmaps 247
Classes Bitmap e BitmapData 249
Manipulação de pixels 251
Cópia de dados de bitmap 254
Compactando dados de bitmap 255
Texturas com funções de ruído 256
Rolagem de bitmaps 258
Benefícios do mapeamento mip 259
Exemplo de bitmap: lua giratória animada 260
Decodificação assíncrona de imagens de bitmap 271

Capítulo 14: Filtro de objetos de exibição
Noções básicas sobre filtragem de objetos de exibição 274
Criação e aplicação de filtros 275
Filtros de exibição disponíveis 282
Exemplo de filtragem de objetos de exibição: bancada do filtro 300

Capítulo 15: Trabalho com sombreadores Pixel Bender
Noções básicas de sombreadores Pixel Bender 308
Carregamento ou incorporação de um sombreador 310
Acesso aos metadados de sombreador 312
Especificação de valores de entrada e de parâmetro de sombreador 313
Uso de um sombreador 318

Capítulo 16: Trabalho com clipes de filme
Noções básicas de clipes de filme 331
Trabalho com objetos MovieClip 332
Controle de reprodução de clipe de filme 332
Criação de objetos MovieClip com o ActionScript 335
Carregamento de um arquivo SWF externo 338
Exemplo de clipe de filme: RuntimeAssetsExplorer 340

Capítulo 17: Trabalho com interpolações de movimento
Noções básicas de interpolações de movimento 344
Cópia de scripts de interpolações de movimento no Flash 345
Incorporação de scripts de interpolações de movimento 346
Descrição da animação 347
Adição de filtros 349
Associação de uma interpolação de movimento com seus objetos de exibição 351

Capítulo 18: Trabalho com cinemática inversa

Noções básicas de cinemática inversa 353
Visão geral da animação de armaduras IK 354
Obtenção de informações sobre uma armadura IK 356
Ocorrência de IKMover e limitação de seu movimento 356
Movimentação de uma armadura IK 357
Utilizando suspensões 358
Uso de eventos IK 358

Capítulo 19: Trabalho em 3D (três dimensões)

Noções básicas de objetos de exibição 3D 360
Noções básicas de objetos de exibição 3D do Flash Player e de tempo de execução do AIR 361
Criação e movimentação de objetos de exibição 3D 363
Projeção de objetos 3D em uma exibição 2D 365
Exemplo: Projeção em perspectiva 367
Execução de transformações 3D complexas 370
Uso de triângulos para obter efeitos 3D 373

Capítulo 20: Princípios básicos do trabalho com texto

Capítulo 21: Uso da classe TextField

Exibição de texto 383
Seleção e manipulação de texto 387
Captura de entrada de texto 389
Restrição de entrada de texto 390
Formatação de texto 391
Renderização avançada de texto 395
Trabalho com texto estático 397
Exemplo de TextField: formatação de texto estilo jornal 398

Capítulo 22: Uso do Flash Text Engine

Criação e exibição de texto 408
Manipulação de eventos no FTE 413
Formatação de texto 416
Trabalho com fontes 420
Controle de texto 423
Exemplo de Flash Text Engine: layout das notícias 429

Capítulo 23: Uso da Text Layout Framework

Visão geral do Text Layout Framework 438
Uso da Text Layout Framework 439
Estruturação de texto com a TLF 444
Formatação de texto com a TLF 448
Importação e exportação de texto com a TLF 449
Gerenciamento de contêineres de texto com a TLF 450
Ativação da seleção de texto, edição e recurso desfazer com a TLF 451
Tratamento de eventos com a TLF 451
Posicionamento de imagens dentro do texto 452

Capítulo 24: Trabalho com som

Noções básicas do trabalho com som 453
Compreensão da arquitetura do som 454
Carregamento de arquivos de som externos 456
Trabalho com sons incorporados 458
Trabalho com arquivos de fluxo de som 460
Trabalho com áudio gerado dinamicamente 461
Reprodução de sons 463
Considerações sobre segurança ao carregar e reproduzir sons 467
Controle do volume e do panorama do som 468
Trabalho com metadados de som 469
Acesso a dados de som brutos 470
Captura de entrada do som 474
Exemplo de som: Podcast Player 480

Capítulo 25: Trabalho com vídeo

Noções básicas sobre vídeo 487
Noções básicas sobre formatos de vídeo 488
Noções básicas sobre a classe Video 492
Carregamento de arquivos de vídeo 493
Controle da reprodução de vídeo 494
Reprodução de vídeo no modo de tela cheia 496
Arquivos de vídeo em fluxo contínuo 500
Noções básicas sobre pontos de sinalização 500
Criação de métodos de retorno de chamada para metadados e pontos de sinalização 501
Uso de pontos de sinalização e metadados 507
Como monitorar a atividade do NetStream 517
Tópicos avançados sobre arquivos de vídeo 520
Exemplo de vídeo: jukebox de vídeo 522
Uso da classe StageVideo para apresentação acelerada por hardware 528

Capítulo 26: Trabalhando com câmeras

Noções básicas sobre a classe Camera 537
Exibição do conteúdo da câmera na tela 538
Desenvolvimento do aplicativo de câmera 538
Conexão à câmera de um usuário 539
Verificação da instalação das câmeras 539
Detecção de permissões de acesso à câmera 540
Maximizando a qualidade de vídeo da câmera 542
Monitorando o status da câmera 543

Capítulo 27: Uso de gerenciamento de direitos digitais

Entendendo o fluxo de trabalho de conteúdo protegido. 545
Membros e eventos relacionados a DRM da classe NetStream 551
Uso da classe DRMStatusEvent 553
Uso da classe DRMAuthenticateEvent 554
Uso da classe DRMErrorEvent 558

Uso da classe DRMManager 558
Uso da classe DRMContentData 560
Atualização do Flash Player para oferecer suporte ao Adobe Access 560
Licenças fora de banda 562
Suporte de domínio 563
Reprodução do conteúdo criptografado usando o suporte de domínio 563
Visualização da licença 564
Entrega do conteúdo 564
Open Source Media Framework 565

Capítulo 28: Inclusão de conteúdo em PDF no AIR
Detecção de recurso de PDF 568
Carregamento de conteúdo em PDF 569
Gravando em script o conteúdo em PDF 569
Limitações conhecidas do conteúdo em PDF no AIR 571

Capítulo 29: Princípios básicos da interação do usuário
Captura da entrada do usuário 573
Gerenciamento do foco 574
Descobrindo tipos de entrada 575

Capítulo 30: Entrada do teclado
Captura da entrada do teclado 577
Uso da classe IME 579
Teclados virtuais 584

Capítulo 31: Entrada do mouse
Captura da entrada do mouse 592
Exemplo de entrada de mouse: WordSearch 595

Capítulo 32: Toque, multitoque e entrada de gestos
Introdução à entrada de toque 600
Descoberta de suporte ao toque 602
Tratamento de eventos de toque 603
Tocar e arrastar 606
Tratamento de Eventos de gestos 608
Solução de problemas 611

Capítulo 33: Copiar e colar
Princípios básicos do copiar e colar 614
Leitura e gravação na área de transferência do sistema 615
Copiar e colar HTML no AIR 615
Formatos de dados da área de transferência 617

Capítulo 34: Entrada do acelerômetro
Verificando suporte ao acelerômetro 624
Detectando mudanças no acelerômetro 624

Capítulo 35: Arrastar e soltar no AIR
Princípios básicos de arrastar e soltar no AIR 627
Suporte ao gesto de arrastar para fora 629
Suporte ao gesto de arrastar para dentro 632
Arrastar e soltar em HTML 635
Arrastar dados para fora de um elemento HTML 639
Arrastar dados para dentro de um elemento HTML 639
Exemplo: Substituição do comportamento padrão de arrastar para dentro HTML 640
Tratamento de arquivos soltos em caixas de proteção HTML de não aplicativo 642
Soltando promessas de arquivo 643

Capítulo 36: Trabalho com menus
Noções básicas do menu 652
Criação de menus nativos (AIR) 659
Sobre menus de contexto em HTML (AIR) 661
Exibição de menus nativos pop-up (AIR) 662
Tratamento de itens de menu 662
Exemplo de menu nativo: menu de janela e de aplicativo (AIR) 664

Capítulo 37: Ícones da barra de tarefas no AIR
Sobre ícones na barra de tarefas 668
Ícones de encaixe 669
Ícones de bandeja do sistema 669
Ícones e botões da barra de tarefas do Windows 671

Capítulo 38: Trabalho com o sistema de arquivos
Uso da classe FileReference 673
Usar a API do sistema de arquivos do AIR 688

Capítulo 39: Armazenamento de dados locais
Objetos compartilhados 724
Armazenamento local criptografado 733

Capítulo 40: Trabalho com bancos de dados SQL locais no AIR
Sobre bancos de dados SQL locais 737
Criação e modificação de um banco de dados 742
Manipulação de dados de um banco de dados SQL 749
Uso de operações de banco de dados síncronas e assíncronas 777
Uso da criptografia com bancos de dados SQL 783
Estratégias para trabalhar com bancos de dados SQL 800

Capítulo 41: Trabalhar com matrizes de bytes
Ler e escrever um ByteArray 803
Exemplo de ByteArray: leitura de um arquivo .zip 809

Capítulo 42: Noções básicas de rede e comunicação
Interfaces de rede 817
Mudanças na conectividade de rede 818
Registro do Sistema de Nome de Domínios (DNS) 821

Capítulo 43: Soquetes
Soquetes TCP ... 823
Soquetes UDP (AIR) ... 834
Endereços IPv6 ... 836

Capítulo 44: Comunicação HTTP
Carregamento de dados externos ... 838
Solicitações do serviço da Web ... 847
Abertura de um URL em outro aplicativo ... 855

Capítulo 45: Comunicação com outras instâncias do Flash Player e AIR
Sobre a classe Local ... 857
Envio de mensagens entre dois aplicativos ... 859
Conexão a conteúdo em domínios diferentes e a aplicativos AIR ... 861

Capítulo 46: Comunicação com processos nativos do AIR
Visão geral das comunicações do processo nativo ... 864
Abertura e fechamento de um processo nativo ... 865
Comunicação com um processo nativo ... 866
Considerações de segurança para a comunicação do processo nativo ... 867

Capítulo 47: Uso da API externa
Noções básicas de uso da API externa ... 869
Requisitos e vantagens da API externa ... 872
Uso da classe ExternalInterface ... 873
Exemplo de API externa: comunicação entre ActionScript e JavaScript em um navegador da Web ... 877

Capítulo 48: Validação da assinatura XML no AIR
Conceitos básicos sobre validação da assinatura XML ... 883
Sobre assinaturas XML ... 888
Implementação da interface IURIDereferencer ... 890

Capítulo 49: Ambiente do sistema cliente
Noções básicas do ambiente do sistema cliente ... 898
Uso da classe System ... 899
Uso da classe Capabilities ... 900
Exemplo de recursos: detecção de capacidades do sistema ... 901

Capítulo 50: Chamada e encerramento do aplicativo do AIR
Chamada do aplicativo ... 906
Captura de argumentos de linha de comando ... 908
Invocando um aplicativo do AIR no logon do usuário ... 911
Invocação de um aplicativo do AIR do navegador ... 912
Encerramento do aplicativo ... 914

Capítulo 51: Trabalho com informações de tempo de execução do AIR e de sistema operacional
Gerenciamento de associações de arquivos ... 916
Obtenção da versão do tempo de execução e do nível de patch ... 917

Detecção de recursos do AIR 917
Rastreamento de presença do usuário 918

Capítulo 52: Trabalho com janelas nativas do AIR
Princípios básicos das janelas nativas no AIR 919
Criação de janelas 926
Gerenciamento de janelas 935
Monitorando eventos de janela 945
Exibição de janelas em tela cheia 946

Capítulo 53: Exibição de telas no AIR
Princípios básicos das telas de exibição no AIR 949
Enumeração das telas 950

Capítulo 54: Impressão
Noções básicas de impressão 953
Impressão de uma página 954
Tarefas do tempo de execução e impressão de sistema do Flash 955
Configuração de tamanho, escala e orientação 958
Técnicas avançadas de impressão 960
Exemplo de impressão: impressão de várias páginas 962
Exemplo de impressão: escala, corte e resposta 964
Exemplo de impressão: configuração de página e opções de impressão 966

Capítulo 55: Geolocalização
Detectando mudanças de geolocalização 969

Capítulo 56: Internacionalização de aplicativos
Introdução à internacionalização de aplicativos 973
Visão geral do pacote flash.globalization 974
Determinando a localidade 975
Formatando números 977
Formatando valores de moeda 979
Formatando data e hora 981
Classificando e comparando strings 983
Conversão de caracteres maiúsculos e minúsculos 985
Exemplo: Internacionalização de um aplicativo de acionamento de ações 986

Capítulo 57: Localização de aplicativos
Seleção de código de idiomas 991
Localização de conteúdo Flex 992
Localização de conteúdo Flash 992
Localização de aplicativos AIR 992
Localização de datas, horas e moedas 993

Capítulo 58: Sobre o ambiente HTML
Visão geral do ambiente HTML 995
AIR e Webkit 998

Capítulo 59: Programação de HTML e JavaScript no AIR

Sobre a classe HTMLLoader 1014
Como evitar erros JavaScript relacionados à segurança 1016
Acesso às classes API do AIR no JavaScript 1021
Sobre URLs no AIR 1022
Como tornar objetos ActionScript disponíveis para JavaScript 1023
Acesso a objetos HTML DOM e JavaScript do ActionScript 1025
Incorporação de conteúdo SWF em HTML 1026
Uso de bibliotecas do ActionScript em uma página HTML 1029
Conversão de objetos Date e RegExp 1031
Manipulação de folha de estilos HTML do ActionScript 1031
Conteúdo cross-scripting em caixas de proteção de segurança distintas 1033

Capítulo 60: Script no contêiner HTML do AIR

Exibição de propriedades de objetos HTMLLoader 1038
Rolagem de conteúdo HTML 1041
Acesso à lista de histórico de HTML 1042
Definição do agente do usuário usado ao carregar conteúdo HTML 1043
Configuração de codificação de caractere para uso de conteúdo HTML. 1043
Definição de interfaces do usuário como navegadores para conteúdo HTML 1044
Criação de subclasses da classe HTMLLoader 1055

Capítulo 61: Tratamento de eventos relacionados a HTML no AIR

eventos HTMLLoader 1057
Tratamento de eventos DOM com o ActionScript 1058
Resposta a exceções JavaScript não capturadas 1058
Tratamento de eventos de tempo de execução com JavaScript 1060

Capítulo 62: Exibição de conteúdo HTML em aplicativos móveis

Objetos StageWebView 1063
Conteúdo 1063
Eventos de navegação 1065
Histórico 1066
Foco 1067
Captura de bitmap 1069

Capítulo 63: Usar workers para simultaneidade

Entendendo workers e simultaneidade 1071
Criando e gerenciando workers 1072
Comunicação entre workers 1075

Capítulo 64: Segurança

Visão geral da segurança da Plataforma Flash 1079
Caixas de proteção de segurança 1081
Controles de permissão 1085
Restrição de APIs de rede 1094
Segurança de modo de tela cheia 1096
Segurança do modo interativo de tela inteira 1097

Carregamento de conteúdo 1098
Cross-scripting 1101
Acesso à mídia carregada como dados 1105
Carregamento de dados 1107
Carregamento de conteúdo incorporado de arquivos SWF importados em um domínio de segurança 1111
Trabalho com conteúdo legado 1111
Configuração de permissões de LocalConnection 1112
Controle do acesso à URL de saída 1112
Objetos compartilhados 1114
Acesso a câmera, microfone, área de transferência, mouse e teclado 1116
Segurança do AIR 1116

Capítulo 65: Como Usar Exemplos do ActionScript
Tipos de exemplos 1139
Execução de exemplos do ActionScript 3.0 no Flash Professional 1141
Execução de exemplos do ActionScript 3.0 no Flash Builder 1142
Execução de exemplos do ActionScript 3.0 em dispositivos móveis 1144

Capítulo 66: Suporte SQL em bancos de dados locais
Sintaxe SQL para a qual há suporte 1149
Suporte ao tipo de dados 1171

Capítulo 67: ID, argumentos e mensagens detalhadas de erro SQL

Capítulo 68: Adobe Graphics Assembly Language (AGAL)
Formato de código de byte AGAL 1179

Capítulo 1: Trabalho com datas e horas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Data e hora talvez não sejam tudo, mas normalmente são um fator chave nos aplicativos de software. O ActionScript 3.0 oferece formas úteis para gerenciar datas de calendário, horas e intervalos de tempo. Duas classes principais permitem gerenciar data e hora: a classe Date e a nova classe Timer do pacote flash.utils.

Datas e horas são informações comuns usadas nos programas ActionScript. Por exemplo, você talvez precise saber o dia da semana atual ou medir o tempo que o usuário gasta em uma tela específica, entre muitas outras possibilidades. No ActionScript, você pode usar a classe Date para representar um único ponto temporal, incluindo informações de data e hora. Em uma ocorrência de Date estão incluídos valores de unidades individuais de data e hora, como ano, mês, data, dia da semana, hora, minutos, segundos, milissegundos e fuso horário. Para usos mais avançados, o ActionScript também inclui a classe Timer, que pode ser usada para realizar ações após um determinado atraso ou em intervalos repetidos.

Mais tópicos da Ajuda

Date

flash.utils.Timer

Gerenciamento de datas de calendário e horas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Todas as funções de gerenciamento de datas de calendário e horas do ActionScript 3.0 estão concentradas na classe Date de nível superior. A classe Date contém métodos e propriedades que permitem manipular datas e horas no horário UTC ou em um fuso horário específico local. O UTC é uma definição de horário padrão basicamente igual ao horário do Meridiano de Greenwich (GMT).

Criação de objetos de Date

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Date tem um dos métodos de construtor mais versáteis de todas as classes básicas. É possível chamá-la de quatro formas diferentes.

Primeiro, se não houver nenhum parâmetro, o construtor `Date()` retornará um objeto de Date que contém data e hora atuais, com base no seu fuso horário local. Eis um exemplo:

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

Segundo, se houver um único parâmetro numérico, o construtor `Date()` irá tratá-lo como o número de milissegundos desde 1º de janeiro de 1970 e retornará um objeto de Date correspondente. Observe que o valor de milissegundos fornecido é tratado como o número de milissegundos desde 1º de janeiro de 1970, em UTC. No entanto, o objeto Date mostra valores no seu fuso horário local, a não ser que você use métodos específicos de UTC para recuperá-los e exibi-los. Se você criar um novo objeto Date usando um único parâmetro de milissegundos, leve em consideração a diferença de fuso entre o horário local e o UTC. As seguintes instruções criam um objeto Date definido como meia-noite do dia 1º de janeiro de 1970, em UTC:

```
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);
```

Terceiro, é possível transmitir vários parâmetros numéricos para o construtor `Date()`. Esses parâmetros são tratados como ano, mês, dia, hora, minuto, segundo e milissegundo, respectivamente, e um objeto Date correspondente é retornado. Supõe-se que esses parâmetros de entrada estão no horário local, não no UTC. As seguintes instruções obtêm um objeto Date definido como meia-noite do dia 1º de janeiro de 2000, no horário local:

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

Quarto, é possível transmitir um parâmetro com uma única string para o construtor `Date()`. O construtor tentará analisar os componentes de data ou hora dessa string e retornará um objeto Date correspondente. Se você usar essa abordagem, é recomendado colocar o construtor `Date()` entre um bloco `try..catch` para detectar qualquer erro de análise. O construtor `Date()` aceita diversos formatos diferentes de sequências de caracteres (que estão listados na Referência do ActionScript® 3.0 para Adobe® Flash® Platform). A instrução a seguir inicializa um novo objeto Date usando um valor de string:

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

Se o construtor `Date()` não conseguir analisar o parâmetro de string, não lançará uma exceção. No entanto, o objeto Date resultante terá um valor de data inválido.

Obtenção de valores de unidade de tempo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode extrair valores para várias unidades de tempo de um objeto Date usando propriedades ou métodos da classe Date. Cada propriedade a seguir fornece o valor de uma unidade de tempo no objeto Date:

- A propriedade `fullYear`
- A propriedade `month`, que está em um formato numérico que vai de 0 para janeiro até 11 para dezembro
- A propriedade `date`, que é o número do calendário do dia do mês, no intervalo de 1 a 31
- A propriedade `day`, que é o dia da semana em formato numérico, com 0 para domingo
- A propriedade `hours`, no intervalo de 0 a 23
- A propriedade `minutes`
- A propriedade `seconds`
- A propriedade `milliseconds`

  Na realidade, a classe Date fornece diversas maneiras para obter cada um desses valores. Você pode obter, por exemplo, o valor do mês de um objeto Date de quatro formas diferentes:

- A propriedade `month`
- O método `getMonth()`

- A propriedade `monthUTC`
- O método `getMonthUTC()`

  Os quatro métodos são praticamente iguais em termos de eficiência, de modo que você pode usar a abordagem mais adequada para seu aplicativo.

  Todas as propriedades listadas acima representam componentes do valor de data total. Por exemplo, a propriedade milliseconds nunca será maior do que 999, pois, quando atingir o número 1000, o valor da propriedade seconds aumentará em uma unidade e a propriedade milliseconds será zerada.

  Se desejar obter o valor do objeto Date em termos de milissegundos desde 1º de janeiro de 1970 (UTC), use o método `getTime()`. Seu correspondente, o método `setTime()`, permite alterar o valor de um objeto Date existente usando os milissegundos desde 1º de janeiro de 1970 (UTC).

Execução de cálculos aritméticos de data e hora

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível realizar operações de adição e subtração em datas e horas com a classe Date. Os valores de Date são mantidos internamente em termos de milissegundos, de modo que você deve converter outros valores em milissegundos antes de adicioná-los ou subtraí-los dos objetos Date.

Se o seu aplicativo efetua vários cálculos aritméticos de data e hora, crie constantes que armazenam valores comuns de unidade de tempo em milissegundos, conforme mostrado a seguir:

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

Agora é fácil efetuar cálculos aritméticos de data usando unidades de tempo padrão. O código a seguir define um valor de data como uma hora do horário atual usando os métodos `getTime()` e `setTime()`:

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

Outra maneira de definir um valor de data é criar um novo objeto Date usando um único parâmetro de milissegundos. Por exemplo, o código a seguir adiciona 30 dias a uma data para calcular outra:

```
// 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));
```

Em seguida, a constante `millisecondsPerDay` é multiplicada por 30 para representar o período de 30 dias e o resultado é adicionado ao valor `invoiceDate` e usado para definir o valor `dueDate`.

Conversão entre fusos horários

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cálculos aritméticos de data e hora são úteis quando você deseja converter datas de um fuso horário para outro. O método `getTimezoneOffset()` serve para isso, pois retorna o valor como a diferença de minutos entre o fuso horário do objeto Date e o UTC. Ele retorna um valor em minutos porque nem todos os fusos horários são definidos como incrementos de horas inteiras - alguns têm diferenças de meia hora em relação às zonas vizinhas.

O exemplo a seguir usa a diferença de fuso para converter uma data do horário local em UTC. Para fazer a conversão, primeiro calcula-se o valor do fuso horário em milissegundos e, depois, ajusta-se o valor de Date de acordo com o resultado:

```
// 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);
```

Controle de intervalos de tempo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao desenvolver aplicativos usando o Adobe Flash CS4 Professional, você tem acesso à linha do tempo, que oferece uma progressão uniforme quadro a quadro do seu aplicativo. No entanto, em projetos exclusivamente do ActionScript, é necessário usar outros mecanismos de controle de tempo.

Loops versus timers

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Em algumas linguagens de programação, você deve desenvolver seus próprios esquemas de controle de tempo usando instruções de loop como `for` ou `do..while`.

As instruções de loop geralmente são executadas conforme permitido pela máquina local, ou seja, o aplicativo é executado mais rapidamente em algumas máquinas e mais lentamente em outras. Se o seu aplicativo precisa de um intervalo de tempo consistente, associe-o a um calendário ou relógio real. Muitos aplicativos, como jogos, animações e controladores em tempo real, precisam de mecanismos de tempo regulares que se adaptem a cada máquina.

A classe Timer do ActionScript 3.0 fornece uma solução incrível. Usando o modelo de eventos do ActionScript 3.0, a classe Timer envia eventos de tempo sempre que um intervalo especificado é atingido.

A classe Timer

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O melhor modo de manipular funções de controle de tempo no ActionScript 3.0 é usar a classe Timer (flash.utils.Timer) para enviar eventos sempre que um intervalo for atingido.

Para iniciar um timer, você precisa criar primeiro uma ocorrência da classe Timer, informando com que frequência um evento de tempo deve ser gerado e quantas vezes isso deve ser feito antes de parar.

Por exemplo, o código a seguir cria uma ocorrência de Timer que envia um evento por segundo e continua durante 60 segundos:

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

O objeto Timer envia um objeto TimerEvent sempre que o intervalo especificado é atingido. Um tipo de evento do objeto TimerEvent é `timer` (definido pela constante `TimerEvent.TIMER`). Um objeto TimerEvent contém as mesmas propriedades de um objeto Event padrão.

Se a ocorrência de Timer for definida como um número fixo de intervalos, um evento `timerComplete` (definido pela constante `TimerEvent.TIMER_COMPLETE`) também será enviado quando o intervalo final for atingido.

Veja um pequeno aplicativo de exemplo que mostra a classe Timer em ação:

```
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!");
        }
    }
}
```

Ao ser criada, a classe ShortTimer cria uma ocorrência de Timer que será acionada uma vez por segundo durante cinco segundos. Em seguida, são adicionados dois ouvintes ao timer: um que ouve cada acionamento e outro que ouve o evento `timerComplete`.

Em seguida, começa o acionamento do timer e, a partir desse ponto, o método `onTick()` é executado em intervalos de um segundo.

O método `onTick()` simplesmente exibe a contagem de acionamentos atual. Depois de cinco segundos, o método `onTimerComplete()` é executado, informando que o tempo acabou.

Ao executar este exemplo, você deve ver as seguintes linhas na janela do console ou rastreamento, na velocidade de uma linha por segundo:

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

Funções de controle de tempo do pacote flash.utils

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 contém várias funções de controle de tempo similares às que estavam disponíveis no ActionScript 2.0. Essas funções são fornecidas no nível do pacote flash.utils e funcionam da mesma maneira como no ActionScript 2.0.

Função	Descrição
`clearInterval(id:uint):void`	Cancela uma chamada `setInterval()` especificada.
`clearTimeout(id:uint):void`	Cancela uma chamada `setTimeout()` especificada.
`getTimer():int`	Retorna o número de milissegundos desde que o Adobe® Flash® Player ou o Adobe® AIR™ foi inicializado.
`setInterval(closure:Function, delay:Number, ... arguments):uint`	Executa uma função em um intervalo especificado (em milissegundos).
`setTimeout(closure:Function, delay:Number, ... arguments):uint`	Executa uma função especificada após um atraso especificado (em milissegundos).

Essas funções estão incluídas no ActionScript 3.0 por questões de compatibilidade com versões anteriores. A Adobe não recomenda utilizá-las em novos aplicativos do ActionScript 3.0. Em geral, é mais fácil e mais eficiente usar a classe Timer em seus aplicativos.

Exemplo de data e hora: relógio analógico simples

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um único exemplo de relógio analógico ilustra esses dois conceitos de data e hora:

- Obtenção de data e hora atuais e extração de valores para horas, minutos e segundos
- Utilização de Timer para definir o ritmo de um aplicativo

  Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo SimpleClock estão localizados na pasta Amostras/SimpleClock. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
SimpleClockApp.mxml ou SimpleClockApp.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/simpleclock/SimpleClock.as	O arquivo de aplicativo principal.
com/example/programmingas3/simpleclock/AnalogClockFace.as	Desenha a superfície de um relógio redondo e os ponteiros de horas, minutos e segundos com base na hora.

Definição da classe SimpleClock

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo do relógio é simples, mas é uma boa ideia para organizar até mesmo aplicativos simples que podem ser expandidos com facilidade no futuro. Para tanto, o aplicativo SimpleClock usa a classe SimpleClock para manipular as tarefas de inicialização e controle de tempo e usa outra classe chamada AnalogClockFace para exibir realmente a hora.

Veja o código que define e inicializa a classe SimpleClock (observe que, na versão Flash, SimpleClock estende a classe Sprite):

```
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;
```

A classe tem duas propriedades importantes:

- A propriedade `face`, que é uma ocorrência da classe AnalogClockFace
- A propriedade `ticker`, que é uma ocorrência da classe Timer

  A classe SimpleClock usa um construtor padrão. O método `initClock()` realiza o trabalho de configuração real, criando a superfície do relógio e iniciando o acionamento da ocorrência de Timer.

Criação da superfície do relógio

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As próximas linhas do código SimpleClock criam a superfície do relógio que é usada para exibir a hora:

```
/**
 * 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();
```

O tamanho da superfície pode ser transmitido para o método `initClock()`. Se nenhum valor de `faceSize` for transmitido, será usado o tamanho padrão de 200 pixels.

Em seguida, o aplicativo inicializa a face e adiciona-a à lista de exibição utilizando o método `addChild()` herdado da classe DisplayObjectContainer. O método `AnalogClockFace.draw()` é chamado para exibir a superfície do relógio uma vez, mostrando a hora atual.

Início do timer

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Assim que a superfície do relógio é criada, o método `initClock()` configura um timer:

```
    // 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();
```

Primeiro, esse método percorre uma ocorrência de Timer que enviará um evento por segundo (a cada 1000 milissegundos). Como nenhum outro parâmetro `repeatCount` é transmitido para o construtor `Timer()`, o Timer será repetido indefinidamente.

O método `SimpleClock.onTick()` será executado uma vez por segundo quando o evento `timer` for recebido:

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

O método `AnalogClockFace.draw()` simplesmente desenha a superfície e os ponteiros do relógio.

Exibição do horário atual

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A maior parte do código na classe AnalogClockFace envolve a configuração dos elementos de exibição da superfície do relógio. Ao ser inicializado, o AnalogClockFace desenha um contorno circular, coloca um rótulo de texto numérico em cada marcação de hora e cria três objetos Shape, um para cada ponteiro do relógio (horas, minutos e segundos).

Assim que o aplicativo SimpleClock é executado, o método `AnalogClockFace.draw()` é chamado a cada segundo do seguinte modo:

```
/**
 * 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);
}
```

Este método salva a hora atual em uma variável para que o horário não mude no meio do desenho dos ponteiros do relógio. Em seguida, o método `showTime()` é chamado para exibir os ponteiros do seguinte modo:

```
/**
 * 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);
}
```

Primeiro, este método extrai os valores para horas, minutos e segundos do horário atual. Depois, esses valores são usados para calcular o ângulo de cada ponteiro. Como faz uma rotação completa em 60 segundos, o ponteiro dos segundos gira 6 graus a cada segundo (360/60). O ponteiro dos minutos gira do mesmo modo em cada minuto.

O ponteiro das horas também é atualizado a cada minuto e pode avançar um pouco à medida que os minutos passam. Ele gira 30 graus por hora (360/12), mas também gira meio grau por minuto (30 graus dividido por 60 minutos).

Capítulo 2: Trabalho com sequências de caracteres

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe String contém métodos que permitem trabalhar com strings de texto. Strings são importantes ao trabalhar com muitos objetos. Os métodos descritos aqui são úteis para trabalhar com sequências de caracteres usadas em objetos como TextField, StaticText, XML, ContextMenu e FileReference.

Strings são sequências de caracteres. O ActionScript 3.0 oferece suporte a caracteres ASCII e Unicode.

Mais tópicos da Ajuda

String

RegExp

parseFloat()

parseInt()

Noções básicas de strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Em linguagem de programação, uma string é um valor de texto, uma sequência de letras, números ou outros caracteres combinados em um único valor. Por exemplo, esta linha de código cria uma variável com o tipo de dados String e atribui um valor de string literal àquela variável:

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

Conforme mostrado nesse exemplo, no ActionScript é possível denotar um valor de string incluindo o texto entre aspas duplas ou simples. Estes são vários exemplos adicionais de strings:

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

Ao manipular uma parte de texto no ActionScript, você está trabalhando com um valor de string. A classe String do ActionScript é o tipo de dados que pode ser usado para trabalhar com valores de texto. As ocorrências de strings são usadas com frequência para propriedades, parâmetros de métodos e assim por diante em muitas outras classes ActionScript.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes relacionados a sequências de caracteres que você encontrará:

ASCII Um sistema que representa caracteres de texto e símbolos em programas de computador. O sistema ASCII oferece suporte ao alfabeto inglês de 26 letras, mais um conjunto limitado de caracteres adicionais.

Caractere A menor unidade de dados de texto (uma única letra ou símbolo).

Concatenação Junção de vários valores de strings com a adição de um ao final do outro, criando um novo valor de string.

String vazia Uma string que não contém nenhum texto, espaço em branco ou outros caracteres, escrita como `""`. Um valor de string vazia é diferente de uma variável String com um valor nulo, uma variável String nula é uma variável que não tem uma ocorrência de String atribuída a ela, enquanto que uma string vazia tem uma ocorrência com um valor que não contém nenhum caractere.

String Um valor textual (sequência de caracteres).

Literal de string (ou "string literal") Um valor de string escrito explicitamente em código como um valor de texto incluído entre aspas duplas ou aspas simples.

Substring Uma string que faz parte de outra string.

Unicode Um sistema padrão que representa caracteres de texto e símbolos em programas de computador. O sistema Unicode permite usar qualquer caractere em qualquer sistema de gravação.

Criação de strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe String é usada para representar dados de string (textuais) no ActionScript 3.0. As strings do ActionScript oferecem suporte a caracteres ASCII e Unicode. A maneira mais simples de criar uma string é usar uma string literal. Para declarar um string literal, use caracteres de aspas duplas retas (`"`) ou de aspas simples (`'`). Por exemplo, as duas strings a seguir são equivalentes:

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

Também é possível declarar uma string usando o operador `new`, da seguinte maneira:

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

As duas strings a seguir são equivalentes:

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

Para usar aspas simples (`'`) em uma string literal definida com delimitadores de aspas simples (`'`), use o caractere de escape barra invertida (`\`). De maneira semelhante, para usar aspas duplas (`"`) em uma string literal definida com delimitadores de aspas duplas (`"`), use o caractere de escape barra invertida (`\`). As duas strings a seguir são equivalentes:

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

É possível escolher o uso de aspas simples ou de aspas duplas com base em quaisquer aspas simples ou duplas existentes em uma string literal, como no exemplo a seguir:

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

Lembre-se de que o ActionScript diferencia aspas simples retas (`'`) e aspas simples esquerda ou direita (`‘` ou `’`). O mesmo é verdadeiro para aspas duplas. Use aspas retas para delinear strings literais. Ao colar texto de outra origem no ActionScript, use os caracteres corretos.

Conforme mostrado na tabela a seguir, é possível usar o caractere de escape de barra invertida (`\`) para definir outros caracteres em strings literais:

Sequência de escape	Caractere
`\b`	Backspace
`\f`	Feed de formulário
`\n`	Nova linha
`\r`	Retorno de carro
`\t`	Tabulação
`\u`***nnnn***	O caractere Unicode com o código de caractere especificado pelo número hexadecimal *nnnn*; por exemplo, `\u263a` é o caractere smiley.
`\\x`***nn***	O caractere ASCII com o código de caractere especificado pelo número hexadecimal *nn*
`\'`	Aspas simples
`\"`	Aspas duplas
`\\`	Caractere de barra invertida simples

A propriedade length

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cada string tem uma propriedade `length` que é igual ao número de caracteres da string:

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

Uma string vazia e uma string nula têm um comprimento de 0, conforme mostrado no exemplo a seguir:

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

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

Trabalho com caracteres em strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cada caractere em uma string tem uma posição de índice na string (um inteiro). A posição do índice do primeiro caractere é 0. Por exemplo, na seguinte string, o caractere `y` está na posição 0 e o caractere `w` está na posição 5:

```
"yellow"
```

É possível examinar caracteres individuais em várias posições em uma string usando os métodos `charAt()` e `charCodeAt()`, como neste exemplo:

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

Quando esse código é executado, a seguinte saída é produzida:

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

Também é possível usar códigos de caracteres para definir uma string usando o método `fromCharCode()`, como no exemplo a seguir:

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

Comparação de strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar os seguintes operadores para comparar strings: `<`, `<=`, `!=`, `==`, `=>` e `>`. Esses operadores podem ser usados com declarações condicionais, como `if` e`while`, como no exemplo a seguir:

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

Ao usar esses operadores com strings, o ActionScript considera o valor de código de cada caractere na string, comparando da esquerda para a direita, como no seguinte exemplo:

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

Use os operadores `==` e `!=` para comparar strings umas com as outras e para comparar strings com outros tipos de objetos, conforme mostrado no exemplo a seguir:

```
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
```

Obtenção de representações de strings de outros objetos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível obter uma representação de String de qualquer tipo de objeto. Todos os objetos têm um método `toString()` para essa finalidade:

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

Ao usar o operador de concatenação `+` com uma combinação de objetos String que não são strings, não é necessário usar o método `toString()`. Para obter detalhes sobre concatenação, consulte a seção a seguir.

A função global `String()` retorna o mesmo valor para um determinado objeto que o valor retornado pelo objeto de chamada do método `toString()`.

Concatenação de strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A concatenação de strings significa utilizar duas strings e uni-las sequencialmente em uma. Por exemplo, é possível usar o operador `+` para concatenar duas strings:

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

Também é possível usar o operador `+=` para produzir o mesmo resultado, conforme mostrado no exemplo a seguir:

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

Além disso, a classe String inclui um método `concat()` que pode ser usado da seguinte maneira:

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

Se você usar o operador `+` (ou o operador `+=`) com o objeto String e um objeto *não*-String, o ActionScript converterá automaticamente o objeto não String em um objeto String para avaliar a expressão, conforme mostrado neste exemplo:

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

No entanto é possível usar parênteses para agrupamento para fornecer contexto para o operador `+`, conforme mostrado no exemplo a seguir:

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

Localização de substrings e padrões em strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Substrings são strings dentro de uma string. Por exemplo, a string `"abc"` tem as seguinte substrings: `""`, `"a"`, `"ab"`, `"abc"`, `"b"`, `"bc"`, `"c"`. É possível usar os métodos do ActionScript para localizar substrings de uma string.

Padrões são definidos no ActionScript por strings ou por expressões regulares. Por exemplo, a expressão regular a seguir define um padrão específico, as letras A, B e C seguidas por um caractere de dígito (as barras são delimitadores de expressões regulares):

```
/ABC\d/
```

O ActionScript inclui métodos para localização de padrões em strings e para substituir correspondências localizadas com substrings de substituição. Esses métodos são descritos nas seções a seguir.

Expressões regulares podem definir padrões intrincados. Para obter mais informações, consulte "Uso de expressões regulares" na página 76.

Localização de uma substring pela posição do caractere

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os métodos `substr()` e `substring()` são semelhantes. Os dois retornam uma substring de uma string. Os dois utilizam dois parâmetros. Nos dois métodos, o primeiro parâmetro é a posição do caractere inicial na string fornecida. No entanto, no método `substr()`, o segundo parâmetro é o *length* da substring a ser retornada e, no método `substring()`, o segundo parâmetro é a posição do caractere no *end* da substring (que não é incluída na string retornada). Este exemplo mostra a diferença entre esses dois métodos:

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

O método `slice()` funciona de maneira semelhante ao método `substring()`. Quando recebe dois inteiros não negativos como parâmetros, ele funciona exatamente da mesma forma. No entanto o método `slice()` pode utilizar inteiros negativos como parâmetros e nesse caso a posição do caractere é utilizada a partir do final da string, conforme mostrado no exemplo a seguir:

```
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
```

É possível combinar inteiros não negativos e negativos como os parâmetros do método `slice()`.

Localização da posição do caractere de uma substring correspondente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar os métodos `indexOf()` e `lastIndexOf()` para localizar substrings dentro de uma string, conforme mostrado no exemplo a seguir:

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

Observe que o método `indexOf()` faz distinção entre maiúsculas e minúsculas.

É possível especificar um segundo parâmetro para indicar a posição do índice na string na qual iniciar a pesquisa, da seguinte maneira:

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

O método `lastIndexOf()` localiza a última ocorrência de uma substring na string:

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

Se você incluir um segundo parâmetro com o método `lastIndexOf()`, a pesquisa será conduzida naquela posição do índice na string operando retroativamente (da direita para a esquerda):

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

Criação de uma matriz de substrings segmentadas por um delimitador

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar o método `split()` para criar uma matriz de substrings que é dividida com base em um delimitador. Por exemplo, é possível segmentar uma string delimitada por vírgula ou por tabulação em várias strings.

O exemplo a seguir mostra como dividir uma matriz em substrings com o caractere e comercial (&) como o delimitador:

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

O segundo parâmetro do método `split()`, que é opcional, define o tamanho máximo da matriz retornada.

Também é possível usar uma expressão regular como o caractere delimitador:

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

Para obter mais informações, consulte "Uso de expressões regulares" na página 76 e a Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Localização de padrões em sequências de caracteres e substituição de subsequências de caracteres

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe String inclui os seguintes métodos para trabalhar com padrões em strings:

- Use os métodos `match()` e `search()` para localizar substrings que correspondem a um padrão.
- Use o método `replace()` para localizar substrings que correspondem a um padrão e substituí-las por uma substring especificada.

Esses métodos são descritos nas seções a seguir.

É possível usar strings ou expressões regulares para definir padrões usados nesses métodos. Para obter mais informações sobre expressões regulares, consulte "Uso de expressões regulares" na página 76.

Localização de substrings correspondentes

O método `search()` retorna a posição do índice da primeira substring que corresponde a um determinado padrão, conforme mostrado neste exemplo:

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

Também é possível usar expressões regulares para definir o padrão a ser correspondido, conforme mostrado no exemplo a seguir:

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

A saída do método `trace()` é 0, porque o primeiro caractere na string é a posição 0 do índice. O sinalizador `i` é definido na expressão regular, portanto a pesquisa não faz distinção entre maiúsculas e minúsculas.

O método `search()` localiza apenas uma correspondência e retorna sua posição inicial no índice, mesmo que o sinalizador `g` (global) esteja definido na expressão regular.

O exemplo a seguir mostra uma expressão regular mais intrincada que corresponde a uma string entre aspas duplas:

```
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.)
```

O método `match()` funciona de maneira semelhante. Ele pesquisa uma substring correspondente. No entanto quando você usa o sinalizador global em um padrão de expressão regular, como no exemplo a seguir, `match()` retorna uma matriz de substrings correspondentes:

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

A matriz `results` é definida como o seguinte:

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

Para obter mais informações sobre expressões regulares, consulte "Uso de expressões regulares" na página 76.

Substituição de substrings correspondentes

É possível usar o método `replace()` para pesquisar um padrão especificado em uma string e substituir correspondências pela string de substituição especificada, conforme mostrado no exemplo a seguir:

```
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.
```

Observe que, neste exemplo, as strings correspondentes não fazem distinção entre maiúsculas e minúsculas porque o sinalizador `i` (`ignoreCase`) está definido na expressão regular, e várias correspondências são substituídas porque o sinalizador `g` (`global`) está definido. Para obter mais informações, consulte "Uso de expressões regulares" na página 76.

É possível incluir os seguintes códigos de substituição `$`na string de substituição. O texto de substituição mostrado na tabela a seguir é inserido no lugar do código de substituição `$`:

$ Code	Texto de substituição
`$$`	$
`$&`	A substring correspondida.
`$` `	A parte da string que precede a substring correspondida. Esse código usa o caractere aspas simples retas esquerdas (`), não aspas simples retas (') ou aspas simples inglesas esquerdas (').
`$'`	A parte da string que segue a substring correspondida. Esse código usa as aspas simples retas (').
$n	A *n*ª correspondência de grupo entre parênteses capturado, em que n é um único dígito, 1 a 9, e $n não é seguido por um dígito decimal.
$nn	A *nn*ª correspondência de grupo entre parênteses capturado, em que *nn* é um número decimal de dois dígitos (01 a 99). Se a *nn*ª captura estiver indefinida, o texto de substituição será uma string vazia.

Por exemplo, veja a seguir o uso dos códigos de substituição `$2` e `$1` que representam o primeiro e o segundo grupos de captura correspondidos:

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

Também é possível usar uma função como o segundo parâmetro do método `replace()`. O texto correspondente é substituído pelo valor retornado da função.

```
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;
}
```

Ao usar uma função como o segundo parâmetro do método `replace()`, os seguintes argumentos são passados para a função:

- A parte correspondente da string.
- Quaisquer correspondências de grupos entre parênteses capturadas. O número de argumentos transmitidos dessa maneira irá variar dependendo do número de correspondências parentéticas. É possível determinar o número de correspondências entre parênteses verificando `arguments.length - 3` no código da função.
- A posição de índice na string em que a correspondência começa.
- A string completa.

Conversão de strings entre maiúsculas e minúsculas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Conforme mostrado no exemplo a seguir, os métodos `toLowerCase()` e `toUpperCase()` convertem caracteres alfabéticos da string em minúsculas e maiúsculas, respectivamente:

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

Após a execução desses métodos, a string de origem permanece inalterada. Para transformar a string de origem, use o seguinte código:

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

Esses métodos funcionam com caracteres estendidos, não simplesmente com a a z e A a Z:

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

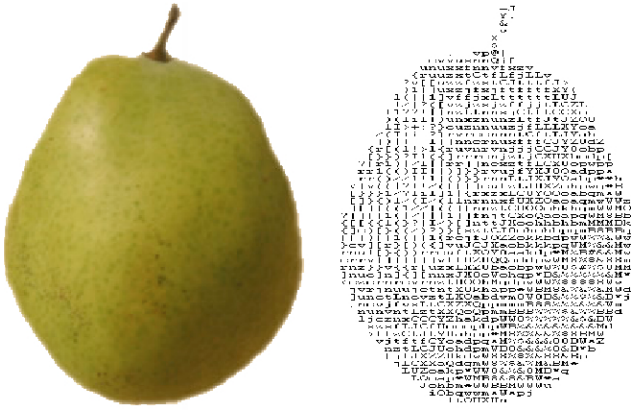
Exemplos de sequências de caracteres: arte ASCII

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Este exemplo de Arte ASCII mostra vários recursos para trabalhar com a classe String no ActionScript 3.0, incluindo o seguinte:

- O método `split()` da classe String é usado para extrair valores de uma string delimitada por caracteres (informações de imagem em um arquivo de texto delimitado por tabulação).
- Várias técnicas de manipulação de string, incluindo `split()`, concatenação e extração de uma parte da string usando `substring()` e `substr()`, são usadas para colocar a primeira letra de cada palavra em maiúscula nos títulos de imagem.
- O método `getCharAt()` é usado para obter um único caractere de uma string (para determinar o caractere ASCII correspondente a um valor de bitmap em escala de cinza).
- A concatenação de string é usada para criar a representação de arte ASCII de uma imagem com um caractere de cada vez.

O termo *arte ASCII* faz referência a representações de texto de uma imagem, na qual uma grade de caracteres de fonte monoespaçada, como caracteres Courier New, plotam a imagem. A imagem a seguir mostra um exemplo de arte ASCII produzida pelo aplicativo:



A versão da arte ASCII do gráfico é mostrada à direita.

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo ASCIIArt podem ser encontrados na pasta Amostras/AsciiArt. O aplicativo consiste nos seguintes arquivos:

Arquivo	**Descrição**
AsciiArtApp.mxml ou AsciiArtApp.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML)
com/example/programmingas3/asciiArt/AsciiArtBuilder.as	A classe que fornece a funcionalidade principal do aplicativo, incluindo a extração de metadados da imagem de um arquivo de texto, carregamento de imagens e gerenciamento do processo de conversão de imagem para texto.
com/example/programmingas3/asciiArt/BitmapToAsciiConverter.as	A classe que fornece o método `parseBitmapData()` para conversão de dados de imagem em uma versão String.
com/example/programmingas3/asciiArt/Image.as	Uma classe que representa uma imagem de bitmap carregada.
com/example/programmingas3/asciiArt/ImageInfo.as	Uma classe que representa metadados para uma imagem arte ASCII (como título, URL do arquivo de imagem, etc.)
imagem/	Uma pasta que contém imagens usadas pelo aplicativo.
txt/ImageData.txt	O arquivo de texto delimitado por tabulação que contém informações sobre as imagens a serem carregadas pelo aplicativo.

Extração de valores delimitados por tabulação

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Este exemplo usa a prática comum de armazenar dados do aplicativo separados do próprio aplicativo. Dessa maneira, se os dados forem alterados (por exemplo, se outra imagem for adicionada ou um título da imagem for alterado) não haverá necessidade de recriar o arquivo SWF. Nesse caso, os metadados da imagem, incluindo o título da imagem, a URL do arquivo real da imagem e alguns valores que são usados para manipular a imagem são armazenados em um arquivo de texto (o arquivo txt\ImageData.ttx do projeto). O conteúdo do arquivo de texto é o seguinte:

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

O arquivo usa um formato delimitado por tabulação específico. A primeira linha é uma linha de título. As linhas restantes contêm os seguintes dados para cada bitmap a ser carregado:

- O nome do arquivo do bitmap.
- O nome de exibição do bitmap.
- Os valores de limite de branco e de limite de preto dos bitmaps. Esses são valores hexadecimais acima e abaixo dos quais um pixel deve ser considerado completamente branco ou completamente preto.

Assim que o aplicativo é iniciado, a classe AsciiArtBuilder carrega e analisa o conteúdo do arquivo de texto para criar a "pilha" de imagens que irá exibir usando o seguinte código do método `parseImageInfo()` da classe AsciiArtBuilder:

```
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);
    }
}
```

Todo o conteúdo do arquivo de texto é contido em uma única ocorrência de String, a propriedade `_imageInfoLoader.data`. Usando o método `split()` com o caractere de nova linha (`"\n"`) como um parâmetro, a ocorrência de String é dividida em uma matriz (`linhas`) cujos elementos são as linhas individuais do arquivo de texto. Em seguida, o código usa um loop para trabalhar com cada uma das linhas (exceto a primeira, porque ela contém apenas cabeçalhos em vez do conteúdo real). Dentro do loop, o método `split()` é usado uma vez novamente para dividir o conteúdo da única linha em um conjunto de valores (o objeto Array denominado `imageProperties`). O parâmetro usado com o método `split()` nesse caso é o caractere de tabulação (`"\t"`), porque os valores de cada linha estão delineados por caracteres de tabulação.

Uso de método String para normalizar títulos de imagens

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma da decisões de design desse aplicativo é que todos os títulos de imagens são exibidos usando um formato padrão, com a primeira letra de cada palavra colocada em maiúscula (exceto por algumas palavras que normalmente não são colocadas em maiúsculas em títulos em inglês). Em vez de assumir que o arquivo de texto contém títulos formatados de maneira apropriada, o aplicativo formata os títulos enquanto eles estão sendo extraídos do arquivo de texto.

Na listagem de código anterior, como parte da extração de valores individuais de metadados da imagem, a seguinte linha de código é usada:

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

Nesse código, o título da imagem do arquivo de texto é passado pelo método `normalizeTitle()` antes de ser armazenado no objeto ImageInfo:

```
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(" ");
}
```

Esse método usa o método `split()` para dividir o título em palavras individuais (separadas pelo caractere espaço), passa cada palavra pelo método `capitalizeFirstLetter()` e, em seguida, usa o método `join()` da classe Array para combinar as palavras em uma única string novamente.

Como o nome sugere, o método `capitalizeFirstLetter()` realmente faz o trabalho de colocar a primeira letra de cada palavra em maiúscula:

```
/**
 * 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;
}
```

Em inglês, o caractere inicial de cada palavra em um título *não* será colocado em maiúscula se a palavra for uma das seguintes: “and”, “the”, “in”, “an”, “or”, “at”, “of” ou “a” (essa é uma versão simplificada das regras). Para executar essa lógica, o código primeiro usa uma declaração `switch` para verificar se a palavra é uma das palavras que não devem ser colocadas em letra maiúscula. Nesse caso, o código simplesmente ignora a declaração `switch`. Por outro lado, se a palavra precisar ser colocada em maiúscula, isso será feito em várias etapas, da seguinte maneira:

1. A primeira letra da palavra é extraída usando `substr(0, 1)` que extrai uma substring a partir do caractere no índice 0 (a primeira letra da string, conforme indicado pelo primeiro parâmetro `0`). A substring terá um caractere de comprimento (indicado pelo segundo parâmetro `1`).
2. Esse caractere é colocado em maiúscula usando o método `toUpperCase()`.
3. Os caracteres restantes da palavra original são extraídos usando `substring(1)` que extrai uma substring no índice 1 (a segunda letra) até o final da string (indicado pela omissão do segundo parâmetro do método `substring()`).
4. A palavra final é criada combinando a primeira letra recém colocada em maiúscula com as letras restantes usando concatenação de strings: `firstLetter + otherLetters`.

Geração do texto de arte ASCII

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe BitmapToAsciiConverter fornece a funcionalidade de converter uma imagem de bitmap em sua representação de texto ASCII. Esse processo é executado pelo método `parseBitmapData()` que é parcialmente mostrado aqui:

```
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;
```

Esse código primeiro define uma ocorrência de String denominada `result` que será usada para criar a versão de arte ASCII da imagem de bitmap. Em seguida, ela executa loop pelos pixels individuais da imagem do bitmap de origem. Usando várias técnicas de manipulação de cores (omitidas aqui para resumir), ela converte os valores das cores vermelho, verde e azul de um pixel individual em um valor de escala de cinza único (um número de 0 a 255). Em seguida, o código divide esse valor por 4 (conforme mostrado) para convertê-lo em um valor na escala de 0 a 63 que é armazenado na variável `index`. (A escala de 0 a 63 é usada porque a "paleta" de caracteres ASCII disponíveis usado por esse aplicativo contém 64 valores.) A paleta de caracteres é definida como uma ocorrência de String na classe BitmapToAsciiConverter:

```
// 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><+_~-;,. ";
```

Como a variável `index` define qual caractere ASCII na paleta corresponde ao pixel atual na imagem de bitmap, esse caractere é recuperado da String `palette` usando o método `charAt()`. Em seguida, ele é anexado à ocorrência da String `result` usando o operador de atribuição de concatenação (`+=`). Além disso, no final de cada linha de pixels, um caractere de nova linha é concatenado ao final da String `result`, forçando uma quebra de linha para criar uma nova linha de "pixels" de caracteres.

Capítulo 3: Trabalho com matrizes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As matrizes permitem que você armazene vários valores em uma única estrutura de dados. É possível usar matrizes indexadas simples que armazenam valores usando índices de números ordinais inteiros e fixos ou matrizes associativas complexas que armazenam valores usando chaves arbitrárias. As matrizes também podem ser multidimensionais, contendo elementos que já são matrizes propriamente ditas. Finalmente, você pode usar um Vetor para uma matriz cujos elementos são ocorrências do mesmo tipo de dados.

Mais tópicos da Ajuda

Matriz

Vetor

Noções básicas sobre matrizes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Em programação é comum o trabalho com uma série de itens, e não com um único objeto. Por exemplo, talvez você queira ter uma lista de músicas pronta para ser reproduzida em um aplicativo de player de música. Você certamente não optaria por criar uma variável específica para cada música da lista. Seria melhor juntar todos os objetos Song em um pacote e trabalhar com eles como se fossem um grupo.

Uma matriz é um elemento de programação que funciona como contêiner para um conjunto de itens, como uma lista de músicas. De modo geral, todos os itens da matriz são ocorrências da mesma classe, mas isso não é uma exigência no ActionScript. Os itens individuais de uma matriz são conhecidos como *elementos* da matriz. A matriz pode ser considerada como um arquivador de variáveis. As variáveis podem ser adicionadas como elementos na matriz, como quando você coloca uma pasta em seu arquivador. Você pode trabalhar com a matriz como uma única variável, como carregar todo o seu arquivo para um outro local. Você pode trabalhar com as variáveis como um grupo, como analisar as pastas uma a uma para buscar informações. Você também pode acessá-las individualmente, como se estivesse abrindo o arquivo e selecionando uma única pasta.

Por exemplo, imagine que está criando um aplicativo de player de música no qual o usuário pode selecionar várias músicas e adicioná-las a uma lista de reprodução. No código do ActionScript, há um método chamado `addSongsToPlaylist()`, que aceita uma única matriz como parâmetro. Independentemente da quantidade de músicas que forem adicionadas à lista (poucas, muitas ou apenas uma), você pode chamar o método `addSongsToPlaylist()` apenas uma vez, transmitindo a matriz que contém os objetos Song. Dentro do método `addSongsToPlaylist()`, você pode usar um loop para navegar entre os elementos da matriz (as músicas) uma a uma e adicioná-las à lista de reprodução.

A *matriz indexada* é o tipo de matriz mais comum do ActionScript. Em uma matriz indexada, cada item é armazenado em um slot numerado (conhecido como *índice*). Os itens são acessados por meio de números, como em endereços. As matrizes indexadas atendem a maior parte das exigências de programação. A classe Array é uma das classes mais comuns utilizada para representar uma matriz indexada.

Em geral, uma matriz indexada é usada para armazenar vários itens do mesmo tipo (objetos que são ocorrências da mesma classe). A classe Array não tem meios de restringir o tipo de itens que ela contém. A classe Vector é o tipo de matriz indexada no qual todos os itens de uma única matriz são do mesmo tipo. O uso da ocorrência Vector em vez de Array também pode proporcionar melhorias no desempenho, além de outras vantagens. A classe Vector está disponível a partir do Flash Player 10 e do Adobe AIR 1.5.

A *matriz multidimensional* representa um caso especial de utilização de matrizes indexadas. Uma matriz multidimensional é uma matriz indexada cujos elementos são matrizes indexadas que, por sua vez, contêm outros elementos.

A *matriz associativa* é um outro tipo de matriz, que usa uma string *key* em vez do índice numérico para identificar elementos individuais. Por fim, o ActionScript 3.0 também possui a classe Dictionary, que representa um *dicionário*. Um dicionário é uma matriz que permite o uso de qualquer tipo de objeto como uma chave de distinção entre elementos.

Conceitos e termos importantes

A lista a seguir de referência contém termos importantes que você vai encontrar ao programar as rotinas de tratamento de matrizes e vetores:

Matriz Um objeto que serve como contêiner para agrupar vários objetos.

Operador de acesso ([]) à matriz Um par de colchetes que circundam um índice ou uma chave e identifica exclusivamente um elemento de matriz. Essa sintaxe é usada após um nome de variável de matriz para especificar um único elemento da matriz, em vez de especificá-la inteira.

Matriz associativa Uma matriz que usa chaves de string para identificar elementos individuais.

Tipo base O tipo de dados dos objetos que uma ocorrência de Vector pode armazenar.

Dictionary Matriz cujos itens consistem em pares de objetos, conhecidos como chaves e valores. A chave é usada no lugar de um índice numérico para identificar um único elemento.

Elemento Um item único de uma matriz.

Índice O "endereço" numérico usado para identificar um único elemento em uma matriz indexada.

Matriz indexada O tipo padrão da matriz que armazena cada elemento em uma posição numerada e usa o número (índice) para identificar elementos individuais.

Tecla A string ou o objeto usado para identificar um único elemento em uma matriz associativa ou em um dicionário.

Matriz multidimensional Matriz que contém itens que são matrizes, em vez de valores únicos.

T A convenção padrão usada nesta documentação para representar o tipo base de uma ocorrência de Vector, independentemente do tipo base. A convenção T é usada para representar um nome de classe, conforme exibido na descrição do parâmetro Type. (“T” corresponde a “tipo”, como em “tipo de dados”)..

Parâmetro de tipo A sintaxe usada com o nome da classe Vector para especificar o tipo base do vetor (o tipo de dados dos objetos que ele armazena). A sintaxe consiste em um ponto (.), seguido do nome do tipo de dados entre colchetes angulares (`<>`). Resumindo, teremos algo como: `Vector.<T>`. Nessa documentação, a classe especificada no parâmetro type é representada genericamente como `T`.

Vetor é um tipo de matriz cujos elementos são todos ocorrências do mesmo tipo de dados.

Matrizes indexadas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As matrizes indexadas armazenam uma série de um ou mais valores organizados de tal maneira que cada valor pode ser acessado com um valor inteiro sem sinal. O primeiro índice é sempre o número 0 e o índice aumenta em incrementos de 1 para cada elemento subsequente adicionado à matriz. No ActionScript 3.0, duas classes são usadas como matrizes indexadas: classes Array e Vector.

As matrizes indexadas usam um inteiro de 32 bits sem sinal como número do índice. O tamanho máximo de uma matriz indexada é 2^{32} - 1 ou 4.294.967.295. Uma tentativa de criar uma matriz maior que o tamanho máximo resulta em um erro de tempo de execução.

Para acessar um elemento individual de uma matriz indexada, use o operador de acesso à matriz (`[]`) para especificar a posição de índice do elemento que deseja acessar. Por exemplo, o código a seguir representa o primeiro elemento (o elemento no índice 0) em uma matriz indexada chamada `songTitles`:

```
songTitles[0]
```

A combinação do nome da variável de matriz seguido pelo índice entre colchetes funciona como um identificador único (em outras palavras, ele pode ser usado da mesma forma que um nome de variável). Você pode atribuir um valor para um elemento de matriz indexada usando o nome e o índice à esquerda de uma instrução de atribuição:

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

Da mesma forma, você pode recuperar o valor de um elemento de matriz indexada usando o nome e o índice à direita de uma instrução de atribuição:

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

Você também pode usar uma variável entre colchetes em vez de fornecer um valor explícito (a variável deve conter um valor inteiro não negativo como uint, um número inteiro positivo ou uma instância de Number de um número inteiro positivo). Geralmente, essa técnica é usada para fazer um loop dos elementos de uma matriz indexada e executar uma operação em alguns ou todos os elementos. A lista de códigos a seguir demonstra essa técnica: O código usa um loop para acessar cada valor em um objeto Array denominado `oddNumbers`. Ele usa a instrução `trace()` para imprimir cada valor da fórmula "oddNumber[*index*] = *value*":

```
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());
}
```

Classe Array

A classe Array é o primeiro tipo de matriz indexada. Uma ocorrência de Array pode armazenar valores de qualquer tipo de dados. O mesmo objeto Array pode armazenar objetos de diferentes tipos de dados. Por exemplo, uma única ocorrência de Array pode ter um valor String no índice 0, uma instância de Number no índice 1 e um objeto XML no índice 2.

Classe Vector

A classe Vector corresponde a outro tipo de matriz indexada que está disponível no ActionScript 3.0. Uma ocorrência de Vector é uma *matriz tipificada*, o que significa que todos os elementos dessa ocorrência possuem o mesmo tipo de dados.

Nota: *A classe Vector está disponível a partir do Flash Player 10 e do Adobe AIR 1.5.*

Quando você declara uma variável Vector ou cria ocorrência de um objeto Vector, está especificando explicitamente os tipos de dados dos objetos que o vetor pode conter. O tipo de dados especificado é conhecido como o *tipo base* do vetor. Nos tempos de execução e de compilação (no modo restrito), qualquer código que define o valor de um elemento Vector ou recupera um valor de um vetor é verificado. Se o tipo de dados do objeto que está sendo adicionado ou recuperado não corresponder ao tipo base do vetor, ocorrerá um erro.

Além da restrição de tipo de dados, a classe Vector tem outras restrições que a diferenciam da classe Array:

- Um vetor é uma matriz densa. Um objeto Array pode ter valores nos índices 0 e 7, mesmo se não tiver valores nas posições de 1 a 6. Entretanto, um vetor deve ter um valor (ou `null`) em cada índice.
- Um vetor pode ter tamanho fixo, opcionalmente. Isso significa que o número de elementos contidos no vetor não pode ser alterado.
- Os limites de acesso aos elementos de um vetor são verificados. Não é possível ler um valor de um índice maior que o elemento final (`tamanho - 1`). Nunca defina um valor com um índice que exceda o índice final atual (em outras palavras, você só pode definir um valor em um índice existente ou no índice `[length]`).

Como resultado das restrições, uma instância Vector tem duas vantagens principais sobre uma instância de Matriz cujos elementos são instâncias de uma única classe:

- Desempenho: a iteração e o acesso ao elemento da matriz são muito mais rápidos ao usar uma ocorrência de Vector do que ao usar uma de Array.
- Segurança do tipo: o compilador pode identificar erros de tipo de dados no modo restrito. Exemplos de erros desse tipo incluem a atribuição de valor de um tipo de dados incorreto ao um vetor ou a espera do tipo de dados errado na leitura de um valor de um vetor. Em tempo de execução, os tipos de dados também são verificados durante a adição ou leitura de dados de um objeto Vector. Observe, entretanto, que ao usar o método `push()` ou `unshift()` para adicionar valores a um vetor, os tipos de dados dos argumentos não são verificados durante a compilação. Quando usar esses métodos, os valores serão verificados no tempo de execução.
- Confiabilidade: a verificação de intervalo de tempo de execução (ou a verificação de comprimento fixo) aumenta a confiabilidade significativamente nas Matrizes.

Com exceção das restrições adicionais e vantagens, a classe Vector é muito parecida com a classe Array. As propriedades e os métodos de um objeto Vector são semelhantes — em grande parte, idênticos — aos de um Array. Na maioria dos casos, quando você usa uma Matriz em que todos os elementos têm o mesmo tipo de dados, uma instância de Vector tem preferência.

Criação de matrizes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar diversas técnicas para criar uma instância de Array ou Vector. No entanto, as técnicas para criação de cada tipo de matriz apresentam diferenças entre si.

Criação de uma ocorrência de Array

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível criar um objeto Array ao chamar o construtor `Array()` ou ao usar a sintaxe literal de matriz.

A função de construtor `Array()` pode ser usada de três modos: Primeiro, se você chama o construtor sem nenhum argumento, obtém uma matriz vazia. Você pode usar a propriedade `length` da classe Array para verificar se a matriz não tem nenhum elemento. Por exemplo, o código a seguir chama o construtor `Array()` sem nenhum argumento:

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

Segundo, se você usar um número como o único parâmetro do construtor `Array()`, uma matriz com esse comprimento será criada, com o valor de cada elemento definido como `undefined`. O argumento deve ser um número inteiro sem sinal entre os valores 0 e 4.294.967.295. Por exemplo, o código a seguir chama o construtor `Array()` com um único argumento numérico:

```
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
```

Terceiro, se você chama o construtor e transmite uma lista de elementos como parâmetros, uma matriz com elementos correspondem a cada parâmetro é criada. O código a seguir transmite três argumentos para o construtor `Array()`:

```
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
```

Você também pode criar matrizes com literais de matriz. Um literal de matriz pode ser atribuído diretamente a uma variável de matriz, como mostra o exemplo a seguir:

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

Criação de uma ocorrência de Vector

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Você pode criar uma ocorrência de Vector ao chamar o construtor `Vector.<T>()`. Você também pode criar um vetor ao chamar a função global `Vector.<T>()`. Essa função converte um objeto especificado em uma ocorrência de Vector. No Flash Professional CS5 e posterior, Flash Builder 4 e posterior, e Flex 4 e posterior, você também pode criar uma instância de vetor usando a sintaxe literal Vector.

Quando você declara uma variável de vetor (ou do mesmo modo, um parâmetro de método de vetor ou um tipo de retorno de método), está especificando o tipo base da variável de vetor. O tipo base também é especificado quando você cria uma ocorrência de Vector ao chamar o construtor `Vector.<T>()`. Em outras palavras, sempre que usar o termo `Vector` no ActionScript, ele estará acompanhado por um tipo base.

O tipo base do vetor é especificado por meio de uma sintaxe de parâmetro de tipo. O parâmetro de tipo segue imediatamente a palavra `Vector` no código. Esse parâmetro consiste em um ponto (.), seguido do nome de classe base entre colchetes angulares (`<>`), conforme apresentado no exemplo a seguir:

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

Na primeira linha do exemplo, a variável `v` está declarada como uma ocorrência de `Vector.<String>`. Em outras palavras, ela representa uma matriz indexada que pode conter apenas ocorrências de String. A segunda linha chama o construtor `Vector()` para criar uma ocorrência do mesmo tipo de vetor, ou seja, um vetor cujos elementos são objetos String. Esse construtor atribui o objeto a `v`.

Uso do construtor Vector.<T>() construtor

Caso você use o construtor `Vector.<T>()` sem quaisquer argumentos, uma ocorrência de Vector vazia será criada. Você pode verificar se um vetor está vazio ao conferir sua propriedade `length`. Por exemplo, o código a seguir chama o construtor `Vector.<T>()` sem argumentos:

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

Você pode predefinir a quantidade de elementos que um vetor precisa ter inicialmente, se tiver essa informação antecipadamente. Para criar um vetor com um determinado número de elementos, transfira o número de elementos como o primeiro parâmetro (o parâmetro `length`). Os elementos Vector são preenchidos com ocorrências do tipo base, já que não podem ficar vazios. Se o tipo base é um tipo de referência que permite valores `null`, todos os elementos conterão `null`. Caso contrário, os elementos conterão o valor padrão para a classe. Por exemplo, uma variável uint não pode ser `null`. Consequentemente, o código a seguir que lista o vetor chamado `ages` é criado com sete elementos contendo o valor 0:

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

Por fim, ao usar o construtor `Vector.<T>()`, você também poderá criar um vetor de tamanho fixo ao passar `true` para o segundo parâmetro (o parâmetro `fixed`). Nesse caso o vetor é criado com o número de elementos especificado e esse número não pode ser alterado. Note, no entanto, que ainda será possível mudar os valores dos elementos de um vetor de tamanho fixo.

Como utilizar o construtor de sintaxe literal Vector

No Flash Professional CS5 e posterior, Flash Builder 4 e posterior e no Flex 4 e posterior, você pode transmitir uma lista de valores ao construtor `Vector.<T>()` para especificar os valores iniciais do vetor:

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

As seguintes informações se aplicam a esta sintaxe:

- A vírgula posterior é opcional.
- Os itens vazios na matriz não são suportados; uma instrução como `var v:Vector.<int> = new <int>[0,,2,]` emite um erro de compilador.
- Você não pode especificar um comprimento padrão da instância Vector. Em vez disso, o comprimento é o mesmo que o número de elementos na lista de inicialização.
- Você não pode especificar se a instância Vector tem um comprimento fixo. Em vez disso, use a propriedade `fixed`.
- A perda de dados ou os erros podem ocorrer se os itens passaram como os valores que não correspondem ao tipo especificado. Por exemplo:

  ```
  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
  ```

Uso do construtor Vector.<T>() função global

Além do construtor `Vector.<T>()` e construtores da sintaxe literal Vector, você também pode usar o `Vector.<T>()` para criar um objeto Vector. A função global `Vector.<T>()` é uma função de conversão. Quando a função global `Vector.<T>()` é chamada, você está especificando o tipo base do vetor que o método retorna. Com isso, uma única matriz indexada (ocorrência de Array ou Vector) é passada como um argumento. Em seguida, o método retorna um vetor com o tipo base especificado, contendo os valores do argumento da matriz de origem. A listagem de código a seguir exibe a sintaxe necessária para chamar a função global `Vector.<T>()`:

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

A função global `Vector.<T>()` executa a conversão do tipo de dados em dois níveis. Primeiro, uma ocorrência de Vector é retornada quando uma ocorrência de Array é passada para a função. Segundo, quando a matriz de origem é uma ocorrência de Array ou Vector, a função tenta converter os elementos da matriz de origem em valores do tipo base. A conversão usa regras de conversão de tipo de dados padrão do ActionScript. Por exemplo, a listagem de código a seguir converte os valores de string na matriz de origem para números inteiros no vetor resultante. A parte decimal do primeiro valor (`"1.5"`) está truncada e o terceiro valor não numérico (`"Waffles"`) é convertido para 0 no resultado:

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

Se não for possível converter nenhum dos elementos de origem, ocorrerá um erro.

Quando o código chama a função global `Vector.<T>()`, se um elemento da matriz de origem é uma ocorrência de uma subclasse do tipo base especificado, o elemento é adicionado ao vetor resultante (não há ocorrência de erros). O uso da função global `Vector.<T>()` é o único modo de converter um vetor com tipo base `T` em um vetor com tipo base que seja uma superclasse de `T`.

Inserção de elementos de matriz

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A maneira mais simples de adicionar um elemento a uma matriz indexada é usar o operador de acesso à matriz (`[]`). Para definir um valor de um elemento de matriz indexada, use o nome de objeto Array ou Vector e o número de índice à esquerda de uma instrução de atribuição:

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

Se a matriz ou o vetor ainda não tiver um elemento no índice, esse índice será criado e o valor será armazenado nele. Se houver um valor no índice, o novo valor substituirá o existente.

Um objeto Array permite a criação de um elemento em qualquer índice. Entretanto, com um objeto Vector você somente poderá atribuir um valor a um índice existente ou ao próximo índice disponível. O próximo índice disponível corresponde à propriedade `length` do objeto Vector. A maneira mais segura de adicionar um novo elemento a um objeto Vector é usar um código como na listagem a seguir:

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

Três métodos da classe Array e Vector - `push()`, `unshift()` e `splice()` - permitem inserir elementos em uma matriz indexada. O método `push()` anexa um ou mais elementos ao final de uma matriz. Em outras palavras, o último elemento inserido na matriz com o método `push()` terá o maior número de índice. O método `unshift()` insere um ou mais elementos no início de uma matriz, sempre no número de índice 0. O método `splice()` insere qualquer número de itens em um índice especificado na matriz.

O exemplo a seguir demonstra os três métodos. Uma matriz chamada `planetas` é criada para armazenar os nomes dos planetas de acordo com a proximidade ao Sol. Primeiro, o método `push()` é chamado para adicionar o item inicial, `Marte`. Segundo, o método `unshift()` é chamado para inserir o item que pertence ao início da matriz, `Mercúrio`. Finalmente, o método `splice()` é chamado para inserir os itens `Vênus` e `Terra` depois de `Mercúrio`, mas antes de `Marte`. O primeiro argumento enviado para `splice()`, o inteiro 1, direciona a inserção para começar no índice 1. O segundo argumento enviado para `splice()`, o inteiro 0, indica que nenhum item deve ser excluído. Finalmente, o terceiro e quarto argumentos enviados para `splice()`, `Vênus` e `Terra`, são os itens que devem ser inseridos.

```
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
```

Os métodos `push()` e `unshift()` retornam um inteiro sem sinal que representa o comprimento da matriz modificada. O método `splice()` retorna uma matriz vazia quando é usado para inserir elementos, o que pode parecer estranho, mas faz mais sentido de acordo com a versatilidade do método `splice()`. Você pode usar o método `splice()` não só para inserir elementos em uma matriz, mas também para remover elementos de uma matriz. Quando usado para remover elementos, o método `splice()` retorna uma matriz que contém os elementos removidos.

Nota: *Se a propriedade `fixed` do objeto Vector for `true`, o número total de elementos do vetor não poderá ser alterado. Se você tentar adicionar um novo elemento a um vetor de tamanho fixo por meio de técnicas aqui descritas, ocorrerá um erro.*

Recuperação de valores e remoção de elementos de matriz

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A maneira mais simples de recuperar o valor de um elemento de uma matriz indexada é usar o operador de acesso à matriz (`[]`). Para recuperar o valor de um elemento de matriz indexada, use o nome de objeto Array ou Vector e o número de índice à direita de uma instrução de atribuição:

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

Você pode tentar recuperar um valor de uma matriz ou vetor usando um índice onde não houver elementos. Nesse caso, um objeto Array retorna um valor indefinido e um vetor emite uma exceção RangeError.

Três métodos das classes Array e Vector - `pop()`, `shift()` e `splice()` - permitem a remoção de elementos. O método `pop()` remove um elemento do final da matriz. Em outras palavras, ele remove o elemento com o número de índice mais alto. O método `shift()` remove um elemento do início da matriz, ou seja, sempre remove o elemento com o número de índice 0. O método `splice()`, que também pode ser usado para inserir elementos, remove um número arbitrário de elementos começando no número de índice especificado pelo primeiro argumento enviado ao método.

O exemplo a seguir usa os três métodos para remover elementos de uma ocorrência de Array. Uma matriz chamada `oceanos` é criada para armazenar os nomes dos grandes corpos de água. Alguns nomes da matriz referem-se a lagos, não a oceanos, e precisam ser removidos.

Primeiro, o método `splice()` é usado para remover os itens `Aral` e `Superior`, e inserir os itens `Atlântico` e `Índico`. O primeiro argumento enviado para `splice()`, o inteiro 2, indica que a operação deve começar com o terceiro item da lista, que está no índice 2. O segundo argumento, 2, indica que dois itens devem ser removidos. Os argumentos restantes, `Atlântico` e `Índico`, são os valores que devem ser inseridos no índice 2.

Segundo, o método `pop()` é usado para remover o último elemento da matriz, `Huron`. Finalmente, o método `shift()` é usado para remover o primeiro item da matriz, `Vitória`.

```
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
```

Os métodos `pop()` e `shift()` retornam o item que foi removido. Para uma ocorrência de Array, o tipo de dados do valor de retorno é Object porque as matrizes podem armazenar valores de qualquer tipo de dados. Para uma ocorrência de Vector, o tipo de dados do valor de retorno é o tipo base do vetor. O método `splice()` retorna uma matriz ou um vetor que contém os valores removidos. Você pode alterar o exemplo de matriz `oceanos` para que a chamada do método `splice()` atribua a matriz retornada a uma nova variável Array, como mostra o exemplo a seguir:

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

Talvez apareça um código que usa o operador `delete` em um elemento de objeto Array. O operador `delete` define o valor de um elemento Array como `undefined`, mas não remove o elemento da matriz. Por exemplo, o código a seguir usa o operador `delete` no terceiro elemento da matriz `oceanos`, mas o comprimento da matriz continua sendo 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
```

Você pode truncar uma matriz ou um vetor usando uma propriedade de matriz `length`. Se a propriedade `length` de uma matriz indexada for definida como um comprimento menor do que o atual, a matriz será truncada, o que remove os elementos armazenados nos números de índice maiores do que o novo valor de `length` menos 1. Por exemplo, se a matriz `oceanos` fosse classificada de modo que todas as entradas válidas estivessem no início da matriz, a propriedade `length` poderia ser usada para remover as entradas no final da matriz, como mostra o código a seguir:

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

Nota: *Se a propriedade `fixed` do objeto Vector for `true`, o número total de elementos do vetor não poderá ser alterado. Se você tentar remover um elemento ou truncar um vetor de tamanho fixo usando as técnicas aqui descritas, ocorrerá um erro.*

Classificação de uma matriz

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Existem três métodos - `reverse()`, `sort()` e `sortOn()` - que permitem alterar a ordem de uma matriz indexada, classificando-a ou invertendo-a. Todos esses métodos modificam a matriz existente. A tabela a seguir resume esses métodos e seus respectivos comportamentos para objetos Array e Vector:

Método	Comportamento de Array	Comportamento de Vector
`reverse()`	Altera a ordem dos elementos de modo de que o último elemento se transforma no primeiro, o penúltimo no segundo e assim por diante.	Idêntico ao comportamento de Array
`sort()`	Permite a classificação dos elementos da matriz de diversos modos predefinidos, como ordem alfabética ou numérica. Também é possível especificar um algoritmo de classificação personalizada.	Classifica os elementos de acordo com o algoritmo de classificação personalizada especificado
`sortOn()`	Permite a classificação de objetos que possuem uma ou mais propriedades em comum, especificando a(s) propriedade(s) que podem ser usadas como chaves de classificação.	Indisponível na classe Vector.

O método reverse()

O método `reverse()` não assume nenhum parâmetro e não retorna um valor, mas permite alternar a ordem da matriz do estado atual para a ordem inversa. O exemplo a seguir inverte a ordem dos oceanos listados na matriz `oceanos`:

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

Classificação básica com o método sort() (somente para a classe Array)

Para uma ocorrência de Array, o método `sort()` reorganiza os elementos de uma matriz usando a *ordem de classificação padrão*. A ordem de classificação padrão deve ter as seguintes características:

- A classificação diferencia maiúsculas de minúsculas, isto é, os caracteres maiúsculos precedem os minúsculos. Por exemplo, a letra D precede a letra b.
- A classificação é crescente, ou seja, os códigos de caracteres menores (como A) precedem os maiores (como B).
- A classificação coloca valores idênticos perto um do outro, mas não em uma ordem específica.
- A classificação baseia-se em strings, ou seja, os elementos são convertidos em strings antes de serem comparados (por exemplo, 10 vem antes de 3 porque a string `"1"` tem um código de caractere menor do que o da string `"3"`).

Talvez você precise classificar uma matriz sem diferenciar maiúsculas de minúsculas, em ordem decrescente ou sua matriz pode conter elementos que devem ser classificados numericamente em vez de em ordem alfabética. O método `sort()` de classe Array tem um parâmetro `options` que permite a alteração de cada característica da ordem de classificação padrão. As opções são definidas por um conjunto de constantes estáticas da classe Array, como mostra a lista a seguir:

- `Array.CASEINSENSITIVE`: essa opção não diferencia maiúsculas de minúsculas durante a classificação. Por exemplo, a letra minúscula b precede a letra maiúscula D.
- `Array.DESCENDING`: essa opção inverte a classificação crescente padrão. Por exemplo, a letra B precede a letra A.
- `Array.UNIQUESORT`: essa opção interrompe a classificação quando dois valores idênticos são encontrados.
- `Array.NUMERIC`: essa opção faz a classificação numérica, de modo que 3 vem antes de 10.

O exemplo a seguir destaca algumas dessas opções. Uma matriz chamada `poetas` é criada e classificada com diversas opções diferentes.

```
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
```

Classificação padrão com o método sort() (somente para as classes Array e Vector)

Além da classificação básica disponível para um objeto Array, também é possível definir uma regra de classificação personalizada. Essa técnica é a única forma do método `sort()` disponível para a classe Vector. Para definir uma classificação padrão, escreva uma função de classificação personalizada e passe-a como um argumento para o método `sort()`.

Por exemplo, se houver uma lista de nomes e cada elemento da lista tiver o nome completo da pessoa, mas você quiser classificar a lista pelo sobrenome, use uma função de classificação personalizada para analisar cada elemento e use o sobrenome na função de classificação. O código a seguir mostra como isso pode ser feito com uma função personalizada que é usada como um parâmetro para o método `Array.sort()`:

```
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
```

A função de classificação personalizada `orderLastName()` usa uma expressão regular para extrair o sobrenome de cada elemento a ser usado para a operação de comparação. O identificador da função `orderLastName` é usado como o único parâmetro ao chamar o método `sort()` na matriz `nomes`. A função de classificação aceita dois parâmetros, `a` e `b`, porque atua em dois elementos de matriz ao mesmo tempo. O valor de retorno da função de classificação indica como os elementos devem ser classificados:

- O valor de retorno -1 indica que o primeiro parâmetro, `a`, precede o segundo, `b`.
- O valor de retorno 1 indica que o segundo parâmetro, `b`, precede o primeiro, `a`.
- O valor de retorno 0 indica que os elementos têm a mesma precedência de classificação.

O método sortOn() (somente para a classe Array)

O método `sortOn()` foi desenvolvido para objetos Array com elementos que contêm objetos. Esses objetos devem ter pelo menos uma propriedade comum que pode ser usada como a chave de classificação. O uso do método `sortOn()` para outros tipos de matriz gera resultados inesperados.

Nota: *A classe Vector não inclui um método `sortOn()`. Esse método está disponível apenas para objetos Array.*

O exemplo a seguir revisa a matriz `poetas` para que cada elemento seja um objeto, em vez de uma string. Cada objeto armazena o sobrenome e o ano de nascimento do poeta.

```
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"});
```

Você pode usar o método `sortOn()` para classificar a matriz pela propriedade `born`. O método `sortOn()` define dois parâmetros, `fieldName` e `options`. O argumento `fieldName` deve ser especificado como uma string. No exemplo a seguir, `sortOn()` é chamado com dois argumentos, `"born"` e `Array.NUMERIC`. O argumento `Array.NUMERIC` é usado para assegurar que a classificação seja feita numericamente, não em ordem alfabética. Isso é útil mesmo quando todos os números têm o mesmo número de dígitos porque garante que a classificação continuará apresentando o comportamento esperado se um número com mais ou menos dígitos for adicionado posteriormente à matriz.

```
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
*/
```

Classificação sem modificação da matriz original (somente para a classe Array)

Geralmente, os métodos `sort()` e `sortOn()` modificam uma matriz. Se desejar classificar uma matriz sem modificar a existente, transmita a constante `Array.RETURNINDEXEDARRAY` como parte do parâmetro `options`. Essa opção instrui os métodos a retornar uma nova matriz que reflete a classificação e deixar a matriz original inalterada. A matriz retornada pelos métodos é uma simples matriz de números de índice que reflete a nova ordem de classificação e não contém nenhum elemento da matriz original. Por exemplo, para classificar a matriz `poetas` por ano de nascimento sem modificá-la, inclua a constante `Array.RETURNINDEXEDARRAY` como parte do argumento transmitido para o parâmetro `options`.

O exemplo a seguir armazena as informações de índice retornadas em uma matriz chamada `índices` e usa a matriz `índices` junto com a matriz `poetas` inalterada para classificar os poetas por ano de nascimento:

```
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
*/
```

Consulta de uma matriz

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quatro métodos das classes Array e Vector - `concat()`, `join()`, `slice()` e `toString()` - consultam a matriz em busca de informações, mas não a modificam. Os métodos `concat()` e `slice()` retornam novas matrizes, enquanto os métodos `join()` e `toString()` retornam strings. O método `concat()` pega uma nova matriz ou lista de elementos como argumentos e a combina com a matriz existente para criar uma nova. O método `slice()` tem dois parâmetros, devidamente chamados de `startIndex` e `endIndex`, e retorna uma nova matriz que contém uma cópia dos elementos "fatiados" a partir da matriz existente. A fatia começa com o elemento em `startIndex` e termina com o elemento bem antes de `endIndex`. A máxima é a mesma: O elemento em `endIndex` não é incluído no valor de retorno.

O exemplo a seguir usa `concat()` e `slice()` para criar novas matrizes com elementos de outras matrizes:

```
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
```

Você pode usar os métodos `join()` e `toString()` para consultar a matriz e retornar seu conteúdo como uma string. Se nenhum parâmetro for usado para o método `join()`, os dois métodos terão o mesmo comportamento: eles retornarão uma string que contém uma lista delimitada por vírgulas de todos os elementos da matriz. O método `join()`, diferente do método `toString()`, aceita um parâmetro chamado `delimiter`, que permite escolher o símbolo a ser usado como separador entre cada elemento na string retornada.

O exemplo a seguir cria uma matriz chamada `rios` e chama `join()` e `toString()` para retornar os valores na matriz como uma string. O método `toString()` é usado para retornar valores separados por vírgula (`riverCSV`), enquanto o método `join()` é usado para retornar valores separados pelo caractere `+`.

```
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
```

Um problema do método `join()` que deve ser levado em consideração é o fato de as ocorrências de Array ou Vector aninhadas sempre serem retornadas com valores separados por vírgula, independentemente do separador especificado para os elementos principais da matriz, como mostra o exemplo a seguir:

```
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
```

Matrizes associativas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma matriz associativa, às vezes chamadas de *hash* ou *mapa*, usa *chaves* em vez de índices numéricos para organizar os valores armazenados. Cada chave de uma matriz associativa é uma string exclusiva que é usada para acessar um valor armazenado. Uma matriz associativa é uma ocorrência da classe Object, o que indica que cada chave corresponde a um nome de propriedade. As matrizes associativas são coleções não ordenadas de pares de chave e valor. Seu código não deve esperar que as chaves de uma matriz associativa estejam em uma ordem específica.

O ActionScript 3.0 também inclui um tipo avançado de matriz associativa chamado *dicionário*. Os dicionários, que são ocorrências da classe Dictionary no pacote flash.utils, usa chaves que podem ser de qualquer tipo de dados. Em outras palavras, as chaves de dicionário não estão limitadas aos valores do tipo String.

Matrizes associativas com chaves de string

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Há duas maneiras de criar matrizes associativas no ActionScript 3.0. A primeira alternativa é usar uma ocorrência de Object. Ao usar essa ocorrência, será possível inicializar a matriz com um literal de objeto. Uma ocorrência da classe Object, também chamada de *objeto genérico*, tem a mesma funcionalidade de uma matriz associativa. Cada nome de propriedade do objeto genérico serve como a chave que fornece acesso a um valor armazenado.

O exemplo a seguir cria uma matriz associativa chamada `monitorInfo`, usando um literal de objeto para inicializar a matriz com dois pares de chave e valor:

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

Se não for necessário inicializar a matriz no tempo da declaração, use o construtor Object para criar a matriz da seguinte maneira:

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

Depois que a matriz é criada com um literal de objeto ou um construtor da classe Object, você pode adicionar novos valores à matriz usando o operador de acesso à matriz (`[]`) ou o operador de ponto (`.`). O exemplo a seguir adiciona dois novos valores a `monitorArray`:

```
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
```

Observe que a chave chamada `aspect ratio` contém um caractere de espaço. Isso é possível com o operador de acesso à matriz (`[]`), mas gera um erro se a tentativa for feita com o operador ponto. Não é recomendado usar espaços em nomes de chave.

A segunda maneira de criar uma matriz associativa é usar o construtor Array (ou o construtor de qualquer classe dinâmica) e usar o operador de acesso à matriz (`[]`) ou o operador de ponto (`.`) para adicionar os pares de chave e valor à matriz. Se declarar que a matriz associativa é do tipo Array, você não poderá usar um literal de objeto para inicializar a matriz. O exemplo a seguir cria uma matriz associativa chamada `monitorInfo` usando o construtor Array e adiciona uma chave chamada `type` e uma chave chamada `resolution`, junto com seus valores:

```
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
```

Não há nenhuma vantagem em usar o construtor Array para criar uma matriz associativa. Você não pode usar a propriedade `Array.length` ou algum método da classe Array com matrizes associativas, mesmo se o construtor Array ou o tipo de dados Array for usado. O uso do construtor Array é mais adequado para a criação de matrizes indexadas.

Matrizes associativas com chaves de objeto (Dicionários)

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar a classe Dictionary para criar uma matriz associativa que usa objetos para chaves em vez de strings. Essas matrizes às vezes são chamadas de dicionários, hashes ou mapas. Pense, por exemplo, em um aplicativo que determina o local de um objeto Sprite com base em sua associação com um recipiente específico. Você pode usar um objeto Dictionary para mapear cada objeto Sprite para um recipiente.

O código a seguir cria três ocorrências da classe Sprite que servem como chaves para o objeto Dictionary. Cada chave recebe o valor `GroupA` ou `GroupB`. Os valores podem ser de qualquer tipo de dados, mas, nesse exemplo, `GroupA` e `GroupB` são ocorrências da classe Object. Posteriormente, você poderá acessar o valor associado a cada chave com o operador de acesso à matriz (`[]`), como mostra o código a seguir:

```
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");
}
```

Iteração com chaves de objeto

É possível percorrer o conteúdo de um objeto Dictionary com um loop `for..in` ou um loop `for each..in`. O loop `for..in` permite percorrer o conteúdo com base nas chaves, enquanto o loop `for each..in` baseia-se nos valores associados a cada chave.

Use o loop `for..in` para acessar diretamente as chaves de um objeto Dictionary. Você também pode acessar os valores do objeto Dictionary com o operador de acesso à matriz (`[]`). O código a seguir usa o exemplo anterior do dicionário `groupMap` para mostrar como percorrer um objeto Dictionary com o loop `for..in`:

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

Use o loop `for each..in` para acessar diretamente os valores de um objeto Dictionary. O código a seguir também usa o dicionário `groupMap` para mostrar como percorrer um objeto Dictionary com o loop `for each..in`:

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

Gerenciamento de memória e chaves de objeto

O Adobe® Flash® Player e o Adobe® AIR™ usam um sistema de coleta de lixo para recuperar a memória que não é mais usada. Quando um objeto não tem nenhuma referência apontada para ele, está qualificado para a coleta de lixo e a memória é recuperada na próxima vez em que o sistema é executado. Por exemplo, o código a seguir cria um novo objeto e atribui uma referência do objeto à variável `myObject`:

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

Contanto que exista alguma referência ao objeto, o sistema de coleta de lixo não recuperará a memória ocupada pelo objeto. Se o valor de `myObject` for alterado de modo que aponte para um objeto diferente ou seja definido como o valor `null`, a memória ocupada pelo objeto original se qualificará para a coleta de lixo, mas somente se não houver nenhuma outra referência ao objeto original.

Se `myObject` for usado como uma chave em um objeto Dictionary, outra referência ao objeto original estará sendo criado. Por exemplo, o código a seguir cria duas referências a um objeto: a variável `myObject` e a chave no objeto `myMap`:

```
import flash.utils.Dictionary;

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

Para que o objeto mencionado por `myObject` se qualifique para a coleta de lixo, remova todas as referências a ele. Nesse caso, altere o valor de `myObject` e exclua a chave `myObject` de `myMap`, como mostra o código a seguir:

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

Como alternativa, você pode usar o parâmetro `useWeakReference` do construtor Dictionary para que todas as chaves de dicionário sejam *referências fracas*. O sistema de coleta de lixo ignora as referências fracas e, desse modo, um objeto que tem apenas referências fracas se qualifica para a coleta de lixo. Por exemplo, no código a seguir, não é necessário excluir a chave `myObject` de `myMap` para que o objeto se qualifique para coleta de lixo:

```
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.
```

Matrizes multidimensionais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As matrizes multidimensionais contêm outras matrizes como elementos. Por exemplo, considere uma lista de tarefas armazenada como uma matriz indexada de strings:

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

Se desejar armazenar uma lista separada de tarefas para cada dia da semana, você poderá criar uma matriz multidimensional com um elemento para cada dia da semana. Cada elemento contém uma matriz indexada, similar à matriz `tarefas`, que armazena a lista de tarefas. É possível usar qualquer combinação de matrizes indexadas ou associativas em matrizes multidimensionais. Os exemplos das seções a seguir usam duas matrizes indexadas ou uma matriz associativa de matrizes indexadas. Se desejar, experimente outras combinações para praticar.

Duas matrizes indexadas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao usar duas matrizes indexadas, você pode visualizar o resultado como uma tabela ou uma planilha. Os elementos da primeira matriz representam as linhas da tabela, enquanto os elementos da segunda matriz representam as colunas.

Por exemplo, a seguinte matriz multidimensional usa duas matrizes indexadas para rastrear as listas de tarefas de cada dia da semana. A primeira matriz, `masterTaskList`, é criada com o construtor da classe Array. Cada elemento da matriz representa um dia da semana; o índice 0 representa segunda-feira e o índice 6 representa domingo. Esses elementos podem ser considerados linhas da tabela. Você pode criar a lista de tarefas de cada dia atribuindo um literal de matriz a cada um dos sete elementos que podem ser criados na matriz `masterTaskList`. Os literais de matriz representam as colunas da tabela.

```
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"];
```

Você pode acessar itens individuais em qualquer lista de tarefas, usando o operador de acesso à matriz (`[]`). O primeiro conjunto de colchetes representa o dia da semana e o segundo representa a lista de tarefas desse dia. Por exemplo, para recuperar a segunda tarefa da lista de quarta-feira, use primeiro o índice 2 para quarta-feira e, em seguida, use o índice 1 para a segunda tarefa da lista.

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

Para recuperar a primeira tarefa da lista de domingo, use o índice 6 para domingo e o índice 0 para a primeira tarefa da lista.

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

Matriz associativa com uma matriz indexada

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para facilitar o acesso a matrizes individuais, você pode usar uma matriz associativa para os dias da semana e uma matriz indexada para as listas de tarefas. A matriz associativa permite usar a sintaxe de ponto ao fazer referência a um dia da semana específico, mas aumenta o tempo de processamento de tempo de execução para acessar cada elemento da matriz associativa. O exemplo a seguir usa uma matriz associativa como base de uma lista de tarefas, com um par de chave e valor para cada dia da semana:

```
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"];
```

A sintaxe de ponto deixa o código mais legível, evitando vários conjuntos de colchetes.

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

É possível percorrer a lista de tarefas usando um loop `for..in`, mas é necessário usar o operador de acesso à matriz (`[]` em vez da sintaxe de ponto para acessar o valor associado a cada chave. Como `masterTaskList` é uma matriz associativa, os elementos não são recuperados necessariamente na ordem esperada, como mostra o exemplo a seguir:

```
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
*/
```

Clonagem de matrizes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Array não tem nenhum método incorporado para fazer cópias de matrizes. Você pode criar uma *cópiasuperficial* de uma matriz chamando os métodos `concat()` ou `slice()` sem nenhum argumento. Em uma cópia superficial, se a matriz original tiver elementos que são objetos, somente as referências aos objetos serão copiadas, não os objetos propriamente ditos. A cópia aponta para os mesmos objetos da original. Qualquer alteração feita nos objetos é refletida nas duas matrizes.

Em uma *cópia profunda*, todos os objetos encontrados na matriz original também são copiados para que a nova matriz não aponte para os mesmos objetos da matriz original. A cópia profunda requer mais de uma linha de código, que normalmente chama uma função de criação. Essa função pode ser criada como uma função do utilitário global ou como um método de uma subclasse de Array.

O exemplo a seguir define uma função chamada `clone()` que faz a cópia profunda. O algoritmo é emprestado de uma técnica comum de programação Java. A função cria uma cópia profunda serializando a matriz em uma ocorrência da classe ByteArray e lendo a matriz de volta em uma nova matriz. Essa função aceita um objeto para que possa ser usada com matrizes indexadas e associativas, como mostra o código a seguir:

```
import flash.utils.ByteArray;

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

Extensão da classe Array

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Array é uma das poucas classes principais que não é final, ou seja, você pode criar sua própria subclasse de Array. Esta seção mostra um exemplo de como criar uma subclasse de Array e discute alguns problemas que podem ocorrer durante o processo.

Como mencionado anteriormente, as matrizes no ActionScript não são tipificadas, mas é possível criar uma subclasse de Array que aceita elementos somente de um tipo de dados específico. O exemplo das seções a seguir define uma subclasse de Array chamada TypedArray que limita seus elementos aos valores do tipo de dados especificado no primeiro parâmetro. A classe TypedArray é apresentada simplesmente como um exemplo que mostra como estender a classe Array e talvez não seja adequada para fins de produção por diversos motivos. Primeiro, a verificação de tipo ocorre durante o tempo de execução, não no tempo de compilação. Segundo, quando um método TypedArray encontra uma discordância, esta é ignorada e nenhuma exceção é lançada, embora os métodos possam ser facilmente modificados para lançar exceções. Terceiro, a classe não pode impedir o uso do operador de acesso à matriz para inserir valores de qualquer tipo na matriz. Quarto, o estilo de codificação favorece a simplicidade por meio da otimização do desempenho.

Nota: *Você pode usar a técnica descrita aqui para criar uma matriz tipificada. No entanto, recomenda-se o uso de um objeto Vector. Uma ocorrência de Vector é uma matriz true type e fornece desempenho e outros aprimoramentos em relação à classe Array ou qualquer subclasse. O objetivo desta discussão é demonstrar como criar uma subclasse Array.*

Declaração da subclasse

Use a palavra-chave `extends` para indicar que uma classe é subclasse de Array. Uma subclasse de Array deve usar o atributo `dynamic`, assim como a classe Array. Caso contrário, a subclasse não funcionará adequadamente.

O código a seguir mostra a definição da classe TypedArray, que contém uma constante para armazenar o tipo de dados, um método de construtor e os quatro métodos que permitem adicionar elementos à matriz. O código de cada método é omitido neste exemplo, mas é descrito e explicado em detalhes nas seções a seguir:

```
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 {}
}
```

Os quatro métodos de substituição usam o espaço para nomes AS3 em vez do atributo `public` porque este exemplo supõe que a opção `-as3` do compilador está definida como `true` e a opção `-es` está definida como `false`. Essas são as configurações padrão do Adobe Flash Builder e do Adobe® Flash® Professional.

Se você for um desenvolvedor experiente que prefere usar protótipos herdados, faça duas pequenas alterações na classe TypedArray para compilá-la com a opção `-es` do compilador definida como `true`. Primeiro, remova todas as ocorrências do atributo `override` e substitua o espaço para nomes AS3 pelo atributo `public`. Segundo, substitua `Array.prototype` nas quatro ocorrências de `super`.

construtor TypedArray

O construtor de subclasse cria um desafio interessante porque aceita uma lista de argumentos de comprimento arbitrário. O desafio agora é transmitir os argumentos para o superconstrutor a fim de criar a matriz. Se você transmitir a lista de argumentos como uma matriz, o superconstrutor irá considerá-la como um único argumento do tipo Array e a matriz resultante terá sempre um elemento. O modo tradicional de manipular a transmissão de listas de argumentos é usar o método `Function.apply()`, que trata uma matriz de argumentos como seu segundo parâmetro, mas a converte em uma lista de argumentos ao executar a função. Infelizmente, o método `Function.apply()` não pode ser usado com construtores.

A única opção restante é recriar a lógica do construtor Array no construtor TypedArray. O código a seguir mostra o algoritmo usado no construtor da classe Array, que pode ser reutilizado no construtor da subclasse de Array:

```
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]
            }
        }
    }
}
```

O construtor TypedArray compartilha a maior parte do código do construtor Array, com apenas quatro alterações no código. Primeiro, a lista de parâmetros inclui um novo parâmetro obrigatório de classe de tipo que permite especificar o tipo de dados da matriz. Segundo, a variável `dataType` é atribuída ao tipo de dados transmitido ao construtor. Terceiro, na instrução `else`, o valor da propriedade `length` é atribuído depois do loop `for` para que `length` inclua somente os argumentos do tipo correto. Quarto, o corpo do loop `for` usa a versão de substituição do método `push()` para que apenas os argumentos do tipo de dados correto sejam adicionados à matriz. O exemplo a seguir mostra a função do construtor TypedArray:

```
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;
        }
    }
}
```

Métodos de substituição de TypedArray

A classe TypedArray substitui os quatro métodos da classe Array que permitem adicionar elementos a uma matriz. Em cada caso, o método de substituição adiciona uma verificação de tipo que impede a adição de elementos que não são do tipo de dados correto. Posteriormente, cada método chama sua própria versão de superclasse.

O método `push()` percorre a lista de argumentos com um loop `for..in` e faz uma verificação de tipo em cada argumento. Todos os argumentos que não são do tipo correto são removidos da matriz `args` com o método `splice()`. Quando o loop `for..in` termina, a matriz `args` contém valores somente do tipo `dataType`. A versão de superclasse de `push()` é chamada com a matriz `args` atualizada, como mostra o código a seguir:

```
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));
}
```

O método `concat()` cria uma matriz TypedArray temporária chamada `passArgs` para armazenar os argumentos aprovados na verificação de tipo. Isso permite a reutilização do código de verificação de tipo existente no método `push()`. O loop `for..in` percorre a matriz `args` e chama `push()` em cada argumento. Como `passArgs` é tipificada como TypedArray, a versão TypedArray de `push()` é executada. Em seguida, o método `concat()` chama sua própria versão de superclasse, como mostra o código a seguir:

```
    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));
    }
```

O método `splice()` usa uma lista arbitrária de argumentos, mas os dois primeiros argumentos sempre fazem referência a um número de índice e ao número de elementos a serem excluídos. É por esse motivo que o método de substituição `splice()` executa a verificação de tipo somente para os elementos da matriz `args` nas posições de índice 2 ou superior. É interessante observar que o código parece ser uma chamada recursiva para `splice()` no loop `for`, mas não é porque `args` é do tipo Array e não TypedArray, o que significa que a chamada para `args.splice()` é uma chamada para a versão de superclasse do método. Quando o loop `for..in` termina, a matriz `args` contém apenas os valores do tipo correto nas posições de índice 2 ou superior e `splice()` chama sua própria versão de superclasse, como mostra o código a seguir:

```
    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));
    }
```

O método `unshift()`, que adiciona elementos ao início de uma matriz, também aceita uma lista arbitrária de argumentos. O método de substituição `unshift()` usa um algoritmo muito parecido com o usado pelo método `push()`, como mostra o exemplo a seguir:

```
    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));
    }
}
```

Exemplo de matriz: PlayList

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo Lista de reprodução demonstra técnicas de trabalho com matrizes, no âmbito de um aplicativo de lista de reprodução que gerencia uma lista de músicas. Estas técnicas são:

- Criação de uma matriz indexada
- Adição de itens a uma matriz indexada
- Classificação de uma matriz de objetos por propriedades diferentes, usando opções de classificação diferentes
- Conversão de uma matriz em uma string delimitada por caracteres

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Lista de reprodução estão localizados na pasta Amostras/Lista de reprodução. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
PlayList.mxml ou PlayList.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/playlist/PlayList.as	Uma classe que representa uma lista de músicas. Usa um Array para armazenar a lista e gerencia a ordenação dos itens da lista.
com/example/programmingas3/playlist/Song.as	O objeto de valor que representa informações sobre uma única música. Os itens gerenciados pela classe PlayList são ocorrências de Song.
com/example/programmingas3/playlist/SortProperty.as	Uma pseudo enumeração cujos valores disponíveis representam as propriedades da classe Song por meio da qual uma lista de objetos Song pode ser classificada.

Visão geral da classe PlayList

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe PlayList gerencia um conjunto de objetos Song. Ela tem métodos públicos e adiciona uma música à lista de reprodução (o método `addSong()`), além de classificar as músicas na lista (o método `sortList()`). Além disso, a classe inclui uma propriedade de acesso somente leitura, `songList`, que fornece acesso ao conjunto real de músicas da lista de reprodução. Internamente, a classe PlayList mantém um registro das músicas usando uma variável de Array privada:

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

Além da variável `_songs` de Array, que é usada pela classe PlayList para manter um registro da lista de músicas, duas outras classes privadas mantém um registro que indica se a lista precisa ser classificada (`_needToSort`) e qual propriedade classifica a lista de músicas em um determinado momento (`_currentSort`).

Assim como com todos os objetos, declarar uma ocorrência de Array é apenas metade do trabalho de criação de uma matriz. Antes de acessar propriedades ou métodos de uma ocorrência de Array, essa classe deve ser instanciada, o que é feito no construtor da classe PlayList.

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

A primeira linha do construtor instancia a variável `_songs`, que fica pronta para ser usada. Além disso, o método `sortList()` é chamado para definir a propriedade sort-by inicial.

Adição de uma música à lista

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o usuário insere uma nova música do aplicativo, o código no formulário de entrada de dados chama o método `addSong()` da classe PlayList.

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

Em `addSong()`, o método `push()` da matriz `_songs` é chamado e adiciona o objeto Song que foi transmitido para `addSong()` como um novo elemento dessa matriz. Com o método `push()`, o novo elemento é adicionado ao final da matriz, independentemente de alguma classificação que tenha sido aplicada anteriormente. Isso significa que, depois que o método `push()` é chamado, a lista de músicas provavelmente não será mais classificada corretamente, de modo que a variável `_needToSort` será definida como `true`. Teoricamente, o método `sortList()` poderia ser chamado imediatamente, não sendo mais necessário manter o registro que indica se a lista deve ou não ser classificada em um determinado momento. No entanto, na prática, não é necessário classificar a lista de músicas imediatamente antes que ela seja recuperada. Adiando a operação de classificação, o aplicativo não executará uma classificação desnecessária se, por exemplo, várias músicas forem adicionadas à lista antes de ser recuperada.

Classificação da lista de músicas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Como as ocorrências de Song gerenciadas pela lista de reprodução são objetos complexos, os usuários do aplicativo talvez queiram classificar a lista de reprodução de acordo com propriedades diferentes, como o título da música ou o ano de publicação. No aplicativo PlayList, a tarefa de classificar a lista de músicas tem três partes: identificar a propriedade que classifica a lista, indicar as opções de classificação que devem ser usadas ao classificar de acordo com essa propriedade e executar a operação de classificação real.

Propriedades de classificação

Um objeto Song mantém o registro de várias propriedades, incluindo o título da música, o artista, o ano de publicação, o nome do arquivo e um conjunto de gêneros definido pelo usuário ao qual a música pertence. Entre essas propriedades, apenas as três primeiras são práticas para classificação. Para facilitar o entendimento dos desenvolvedores, o exemplo inclui a classe SortProperty, que age como uma enumeração com valores que representam as propriedades disponíveis para classificação.

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

A classe SortProperty contém três constantes (`TITLE`, `ARTIST` e `YEAR`), que armazenam uma string com o nome real da propriedade associada da classe Song que pode ser usada para classificação. No restante do código, sempre que uma propriedade de classificação for indicada, o membro de enumeração será usado. Por exemplo, no construtor PlayList, a lista é classificada inicialmente chamando o método `sortList()` do seguinte modo:

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

Como a propriedade de classificação é especificada como `SortProperty.TITLE`, as músicas são classificadas de acordo com o título.

Classificação por propriedade e especificação das opções de classificação

A classificação real da lista de músicas é realizada pela classe PlayList no método `sortList()` do seguinte modo:

```
/**
 * 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;
}
```

Quando a classificação é feita por título ou artista, faz sentido classificar em ordem alfabética, mas, quando a classificação é feita por ano, é mais lógico realizar uma classificação numérica. A instrução `switch` é usada para definir a opção de classificação adequada, armazenada na variável `sortOptions`, de acordo com o valor especificado no parâmetro `sortProperty`. Aqui, mais uma vez, os membros de enumeração nomeados são usados para diferenciar as propriedades, em vez de valores codificados.

Quando a propriedade e as opções de classificação são determinadas, a matriz `_songs` é realmente classificada chamando o método `sortOn()` e transmitindo esses dois valores como parâmetros. A propriedade de classificação atual é registrada, o que indica que a lista de músicas está classificada no momento.

Combinação de elementos de matriz em uma string delimitada por caracteres

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Além de usar uma matriz para manter a lista de músicas na classe PlayList, neste exemplo, as matrizes também são usadas na classe Song para ajudar a gerenciar a lista de gêneros aos quais pertencem as músicas. Veja este snippet da definição da classe Song:

```
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(";");
}
```

Ao criar uma nova ocorrência de Song, o parâmetro `genres` que é usado para especificar os gêneros das músicas é definido como uma ocorrência de Array. Desse modo, é fácil agrupar vários gêneros em uma única variável que pode ser transmitida para o construtor. No entanto, internamente, a classe Song mantém os gêneros na variável `_genres` privada como uma ocorrência de String separada por ponto-e-vírgula. O parâmetro Array é convertido em uma string separada por ponto-e-vírgula chamando seu método `join()` com o valor de string literal `";"` como o delimitador especificado.

Com o mesmo token, os acessadores de `genres` permitem que os gêneros sejam definidos ou recuperados como uma matriz:

```
    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(";");
    }
```

O acessador `genresset` se comporta exatamente como o construtor: aceita uma matriz e chama o método `join()` para convertê-la em uma string separada por ponto-e-vírgula. O acessador `get` executa a operação oposta: o método `split()` da variável `_genres` é chamado, dividindo a string em uma matriz de valores que usa o delimitador especificado (o valor de string literal `";"` como antes).

Capítulo 4: Manipulação de erros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

"Tratar" um erro significa incluir lógica em seu aplicativo para reagir a um erro ou solucioná-lo. Os erros são gerados quando um aplicativo é compilado ou quando um aplicativo compilado estiver em execução. Quando o aplicativo manipula erros e o erro é encontrado, acontece *algo* em resposta, em vez de ficar sem resposta (quando qualquer processo que criou o erro falha discretamente). Usado corretamente, a manipulação de erros ajuda a proteger o aplicativos e seus usuários de um comportamento inesperado.

No entanto, o tratamento de erros é uma ampla categoria que inclui responder a muitos tipos de erros que são lançados durante a compilação ou enquanto um aplicativo está em execução. Essa discussão concentra-se em como tratar erros em tempo de execução (gerados durante a execução de um aplicativo), nos tipos diferentes de erros que podem ser gerados e nas vantagens do sistema de tratamento de erros no ActionScript 3.0.

Noções básicas da manipulação de erros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O erro em tempo de execução é algo que dá errado no seu código de ActionScript e que impede a execução do conteúdo do ActionScript da forma pretendida. Para garantir que o seu código do ActionScript seja executado corretamente para os usuários, você deve escrevê-lo no aplicativo que manipula o erro (que o corrige, contorna ou pelo menos deixa o usuário saber o que aconteceu). Esse processo é chamado de *manipulação de erros*.

O tratamento de erros é uma ampla categoria que inclui responder a muitos tipos de erros que são lançados durante a compilação ou enquanto um aplicativo está em execução. Os erros que acontecem em tempo de compilação, em geral, são mais fáceis de identificar; é necessário corrigi-los para concluir o processo de criação de um arquivo SWF.

Os erros de tempo de execução podem ser mais difíceis de detectar, porque, para que ocorram, o código com falha deve realmente ser executado. Se um segmento do seu programa tiver várias ramificações de código, como uma instrução `if..then..A instrução else` testa cada condição possível, com todos os valores de entrada possíveis que os usuários reais podem usar, para confirmar que o seu código não possui erros.

Os erros de tempo de execução podem ser divididos em duas categorias: *erros de programa* são erros no seu código do ActionScript, como especificar o tipo de dados errado para o parâmetro de um método; *erros lógicos* são erros na lógica (na verificação de dados e manipulação de valores) do seu programa, tal como usar a fórmula errada para calcular taxas de juros em um aplicativo bancário. Novamente, os dois tipos de erros, em geral, podem ser detectados e corrigidos com antecedência, testando prontamente o aplicativo.

Provavelmente, você desejará identificar e remover todos os erros do aplicativo antes de lançá-lo para os usuários finais. Entretanto, nem todos os erros podem ser previstos ou evitados. Por exemplo, suponha que o aplicativo do ActionScript carregue informações de um site específico que esteja fora do seu controle. Se, em algum momento, esse site não estiver disponível, a parte do seu aplicativo que depende desses dados externos não se comportará corretamente. O aspecto mais importante do tratamento de erros diz respeito à preparação para os casos desconhecidos ao tratamento desses erros de maneira elegante. Os usuários precisam continuar usando seu aplicativo, ou pelo menos ver uma mensagem de erro intuitiva explicando o motivo do defeito.

Os erros de tempo de execução são representados de duas formas no ActionScript:

- Classes de erro: muitos erros possuem uma classe de erro associada. Quando ocorre um erro, o tempo de execução do Flash (como o Flash Player ou o Adobe AIR) cria uma ocorrência da classe de erro específica associada a esse erro em particular. O seu código pode usar as informações contidas nesse objeto de erro para dar uma resposta apropriada ao erro.
- Eventos de erro: às vezes ocorre um erro quando o tempo de execução do Flash normalmente dispararia um evento. Nesses casos, é disparado um evento de erro. Cada evento de erro possui uma classe associada, e o tempo de execução do Flash transmite uma instância dessa classe para os métodos que são inscritos no evento de erro.

Para determinar se um método em particular pode disparar um erro ou um evento de erro, consulte a entrada do método na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Conceitos e termos importantes

A seguinte lista de consulta contém termos importantes para a programação de rotinas de tratamento de erros:

Assíncrono Um comando de programa, como uma chamada de método, que não fornece um resultado imediato e, em vez disso, apresenta um resultado (ou erro) na forma de um evento.

Catch quando ocorre uma exceção (um erro de tempo de execução) e seu código obtém essa informação, ele recebe a instrução *catch* para detectar a exceção. Depois que uma execução é capturada, o tempo de execução do Flash para de notificar outro código do ActionScript da exceção.

Versão do deputador Uma versão especial do tempo de execução do Flash, como a versão de depuração do Flash Player do AIR Debug Launcher (ADL), que contém código para notificar os usuários dos erros em tempo de execução. Na versão padrão do Flash Player ou do Adobe AIR (aquela que a maioria dos usuários possui), os erros que não são manipulados pelo código do ActionScript são ignorados. Nas versões de depurador (que são incluídas no Adobe Flash CS4 Professional e no Adobe Flash Builder), é exibida uma mensagem de aviso quando acontece um erro não manipulado.

Exceção Um erro que ocorre enquanto um aplicativo está sendo executado e que o tempo de execução do Flash não pode resolver sozinho.

Relançar Quando seu código captura uma exceção, o tempo de execução do Flash não notifica mais outros objetos da exceção. Se for importante que os outros objetos recebam a exceção, o seu código deverá *gerar novamente* a exceção para reiniciar o processo de notificação.

Síncrono Um comando de programa, como uma chamada de método, que fornece um resultado imediato (ou imediatamente gera um erro), o que significa que a resposta pode ser usada dentro do mesmo bloco de código.

Lançar O ato de notificar o tempo de execução do Flash (e consequentemente notificar outros objetos e código do ActionScript) de que um erro ocorreu é conhecido como *lançar* um erro.

Tipos de erros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando desenvolver e executar aplicativos, você encontrará diferentes tipos de erros e terminologia de erros. A lista a seguir apresenta os principais termos e tipos de erros:

- *Erros de tempo de compilação* são gerados pelo compilador do ActionScript durante a compilação de código. Os erros de compilação ocorrem quando problemas sintáticos no seu código impedem a criação do aplicativo.

- *Erros de tempo de execução* ocorrem quando você executa o aplicativo depois de compilá-lo. Os erros de tempo de execução representam os erros causados quando um arquivo SWF é reproduzido no tempo de execução do Flash (como Adobe Flash Player ou Adobe AIR). Na maioria dos casos, você poderá manipular os erros de tempo de execução assim que eles ocorrerem, relatando-os ao usuário e tomando medidas para manter o aplicativo em execução. Se o erro for fatal, por exemplo, não for possível conectar a um site remoto ou carregar os dados necessários, você poderá usar a manipulação de erros para permitir que o aplicativo seja concluído normalmente.
- Os *erros síncronos* são erros de tempo de execução que ocorrem no momento em que uma função é chamada; por exemplo, quando você tenta usar um método específico e o argumento que transmite para o método é inválido, o tempo de execução do Flash gera uma exceção. A maioria dos erros ocorre de forma síncrona, no momento em que a instrução é executada, e o fluxo de controle transmite imediatamente para a instrução `catch` mais aplicável.

  Por exemplo, o trecho de código a seguir lança um erro de tempo de execução porque o método `browse()` não é chamado antes de o programa tentar carregar um arquivo:

  ```
  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.
  }
  ```

  Nesse caso, um erro de tempo de execução é lançado de forma síncrona porque o Flash Player determinou que o método `browse()` não foi chamado antes da tentativa de upload do arquivo.

  Para obter informações detalhadas sobre manipulação de erros síncronos, consulte "Manipulação de erros síncronos em um aplicativo" na página 58.

- *Errosassíncronos* são erros de runtime que ocorrem fora do fluxo do programa. Eles geram eventos e os ouvintes de eventos os capturam. Uma operação assíncrona é aquela na qual uma função inicia uma operação, mas não espera ela ser concluída. Você pode criar um ouvinte de eventos de erro que espere o aplicativo ou usuário tentar alguma operação e, se a operação falhar, detecte o erro com um ouvinte de eventos e responda ao evento de erro. Depois, o ouvinte de eventos chama uma função de manipulador de eventos para responder ao evento de erro da melhor maneira. Por exemplo, o manipulador de eventos pode iniciar uma caixa de diálogo que solicite que o usuário resolva o erro.

  Considere o exemplo de erro síncrono de carregamento de arquivo apresentado anteriormente. Se você chamar com êxito o método `browse()` antes de começar a carregar um arquivo, o Flash Player irá despachar vários eventos. Por exemplo, quando um upload é iniciado, o evento `open` é despachado. Quando a operação de upload de arquivo é concluída com êxito, o evento `complete` é despachado. Como a manipulação de eventos é assíncrona (isto é, não ocorre em momentos específicos, conhecidos e predeterminados), é necessário usar o método `addEventListener()` para ouvir esses eventos específicos, como mostra o seguinte código:

```
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...");
}
```

Para obter informações detalhadas sobre manipulação de erros assíncronos, consulte "Resposta a eventos e status de erros" na página 63.

- *Exceções não detectadas* são erros lançados sem nenhuma lógica correspondente (como uma instrução `catch`) em resposta a elas. Se o seu aplicativo lançar um erro, e nenhuma instrução `catch` apropriada ou manipulador de eventos puderem ser encontrados no nível atual ou superior para manipular o erro, o erro será considerado uma exceção não detectada.

  Quando ocorrer um erro não capturado, o aplicativo gera um erro `uncaughtError`. Esse evento também é conhecido como "tratamento global de erros". Este evento é gerado pelo objeto UncaughtErrorEvents do SWF, que está disponível através da propriedade `LoaderInfo.uncaughtErrorEvents`. Se não houver nenhum ouvinte registrado para o evento `uncaughtError`, o aplicativo ignorará os erros não capturados e tentará continuar a execução, enquanto o erro não fizer o SWF parar.

  Além de gerar o evento `uncaughtError`, as versões de depurador do tempo de execução do Flash reagem aos erros não capturados por meio do encerramento do script atual. Então, elas exibem o erro não capturado na saída do comando `trace` ou gravando a mensagem de erro em um arquivo de log. Se o objeto de exceção for uma instância da classe Error ou de uma de suas subclasses, as informações de rastreamento de pilha também serão recebidas na saída. Para obter mais informações sobre como usar a versão de depurador dos tempos de execução do Flash, consulte "Trabalho com as versões de depurador dos tempos de execução do Flash" na página 57.

  Nota: *Durante o processamento de um evento uncaughtError, se um evento de erro for lançado por um manipulador uncaughtError, o manipulador de evento é chamado várias vezes. Isso resulta em um loop infinito de exceções. Recomenda-se evitar essa situação.*

Manipulação de erros no ActionScript 3.0

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Como muitos aplicativos podem ser executados sem construir a lógica para manipular erros, os desenvolvedores ficam tentados a adiar a criação da manipulação de erros em seus aplicativos. Entretanto, sem isso, um aplicativo pode parar facilmente ou frustrar o usuário caso algo não funcione conforme o esperado. O ActionScript 2.0 possui uma classe Error que permite criar a lógica dentro de funções personalizadas e lançar uma exceção com uma mensagem específica. Como a manipulação de erros é essencial para tornar um aplicativo amigável, o ActionScript 3.0 inclui uma arquitetura expandida para detectar erros.

Nota: *Embora a Referência do ActionScript® 3.0 para Adobe® Flash® Platform documente as exceções geradas por muitos métodos, talvez ela não contemple todas as exceções possíveis de cada método. Um método pode gerar uma exceção para erros de sintaxe ou outros problemas que não são observados explicitamente na descrição do método, mesmo quando a descrição lista algumas das exceções que um método gera.*

Elementos de manipulação de erros do ActionScript 3.0

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 inclui várias ferramentas para manipulação de erros, incluindo:

- Classes de erro. O ActionScript 3.0 inclui uma ampla variedade de classes Error para expandir o escopo de situações que podem produzir objetos de erro. Cada classe Error ajuda os aplicativos a manipular e responder a condições de erro específicas, quer estejam relacionadas a erros de sistema (como uma condição MemoryError), erros de código (como uma condição ArgumentError), erros de rede e comunicação (como uma condição URIError) ou a outras situações. Para obter mais informações sobre cada classe, consulte "Comparação das classes Error" na página 67.
- Menos falhas silenciosas. Nas versões anteriores do Flash Player, os erros eram gerados e relatados somente se você usasse a instrução `throw` explicitamente. Para o Flash Player 9 e tempos de execução do Flash posteriores, métodos e propriedades originais do ActionScript geram erros de tempo de execução. Esses erros permitem que você trate essas exceções de maneira mais eficaz quando elas ocorrem, depois reaja individualmente a cada exceção.
- Limpar mensagens de erro exibidas durante a depuração. Ao usar a versão de depurador de um tempo de execução do Flash, situações ou códigos problemáticos geram mensagens de erro robustas, o que ajuda a identificar facilmente os motivos da falha de um bloco de código específico. Essas mensagens tornam mais eficaz a correção dos erros. Para obter mais informações, consulte "Trabalho com as versões de depurador dos tempos de execução do Flash" na página 57.
- Erros precisos permitem a exibição de claras mensagens de erro aos usuários. Nas versões anteriores do Flash Player, o método `FileReference.upload()` retornava um valor booliano de `false` se a chamada `upload()` não fosse bem-sucedida, indicando um de cinco erros possíveis. Se ocorrer um erro ao chamar o método `upload()` no ActionScript 3.0, quatro erros específicos ajudam a exibir mensagens de erro mais precisas aos usuários finais.
- Manipulação de erros refinada. Erros distintos são lançados para várias situações comuns. Por exemplo, no ActionScript 2.0, antes de um objeto FileReference ser preenchido, a propriedade `name` tem o valor `null` (portanto, antes de usar ou exibir a propriedade `name`, é necessário garantir que o valor seja definido e não seja `null`). No ActionScript 3.0, se você tentar acessar a propriedade `name` antes que ela seja preenchida, o Flash Player ou o AIR lançará IllegalOperationError, informando que o valor não foi definido, e você pode usar blocos `try..catch..finally` para manipular o erro. Para obter mais informações, consulte, "Uso das instruções try..catch..finally" na página 58.

- Nenhuma desvantagem de desempenho significativa. O uso de blocos `try..catch..finally` para manipular erros exige poucos (ou nenhum dos) recursos adicionais comparado às versões anteriores do ActionScript.
- Uma classe ErrorEvent que permite criar ouvintes para eventos de erro assíncronos específicos. Para obter mais informações, consulte "Resposta a eventos e status de erros" na página 63.

Estratégias de manipulação de erros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Contanto que o aplicativo não encontre uma condição problemática, ele poderá ser executado com êxito se você não criar uma lógica de manipulação de erros no seu código. Entretanto, se você não manipular os erros ativamente e seu aplicativo encontrar um problema, seus usuários nunca saberão por que ocorreu uma falha.

Há diferentes maneiras de abordar a manipulação de erros no seu aplicativo. A lista a seguir resume as três principais opções para manipular erros:

- Usar instruções `try..catch..finally`. Esses comandos detectarão os erros síncronos que ocorrerem. Você pode aninhar essas instruções em uma hierarquia para detectar exceções em diversos níveis de execução de código. Para obter mais informações, consulte, "Uso das instruções try..catch..finally" na página 58.
- Criar seus próprios objetos de erro personalizados. Você pode usar a classe Error para criar seus próprios objetos de erro personalizados a fim de controlar operações específicas no seu aplicativo que não estejam incluídas nos tipos de erro embutidos. Depois, você pode usar as instruções `try..catch..finally` nos seus objetos de erro personalizados. Para obter mais informações, consulte "Criação de classes de erros personalizadas" na página 62.
- Escreva ouvintes e manipuladores de eventos para responder aos eventos de erro. Usando essa estratégia, você pode criar manipuladores de erro globais que permitem manipular eventos semelhantes sem duplicar uma grande quantidade de código em `try.catch..finally`. Você tem muito mais chances de detectar erros assíncronos usando essa abordagem. Para obter mais informações, consulte "Resposta a eventos e status de erros" na página 63.

Trabalho com as versões de depurador dos tempos de execução do Flash

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A Adobe fornece aos desenvolvedores edições especiais dos tempos de execução do Flash para ajudar nos esforços de depuração. Ao instalar o Adobe Flash Professional ou o Adobe Flash Builder, você obtém uma cópia da versão de depurador do Flash Player. Ao instalar uma dessas ferramentas, você obtém um utilitário para a depuração dos aplicativos Adobe AIR, também chamado de ADL, também fornecida como parte do Adobe AIR SDK.

Há uma grande diferença na forma como as versões de depurador e de lançamento do Flash Player e do Adobe AIR indicam os erros. As versões de depurador mostram o tipo de erro (como Error, IOError ou EOFError genéricos), o número de erros e uma mensagem de erro legível. As versões de lançamento mostram apenas o tipo de erro e o número de erros. Por exemplo, considere o seguinte código:

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

Se o método `readBoolean()` gerar um EOFError na versão de depurador do Flash Player, a seguinte mensagem será exibida no campo de texto `tf`: "EOFError: Erro 2030: Fim do arquivo localizado."

O mesmo código em uma versão de lançamento do Flash Player ou do Adobe AIR exibiria o seguinte texto: "EOFError: Erro 2030."

Nota: *Os players depuradores transmitem um evento chamado "allComplete"; evite criar eventos personalizados com o nome "allComplete". Caso contrário, você encontrará comportamentos imprevisíveis ao depurar.*

Para manter os recursos e o tamanho mínimos nas versões de lançamento, as sequências de caracteres de mensagem de erro não estão presentes. Você pode pesquisar o número do erro na documentação (os apêndices da Referência do ActionScript® 3.0 para Adobe® Flash® Platform) para correlacionar a uma mensagem de erro. Se preferir, você pode reproduzir o erro usando as versões de depurador do Flash Player e do AIR para ver a mensagem inteira.

Manipulação de erros síncronos em um aplicativo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A manipulação de erros mais comum é a lógica de manipulação de erros síncrona, em que você insere instruções no código para detectar erros síncronos enquanto um aplicativo está em execução. Esse tipo de manipulação de erros permite ao aplicativo notar e se recuperar de erros de tempo de execução quando as funções falham. A lógica para detectar um erro síncrono inclui as instruções `try..catch..finally`, que literalmente tentam uma operação, detectam qualquer resposta de erro do tempo de execução do Flash e finalmente executam outra operação para manipular a operação que falhou.

Uso das instruções try..catch..finally

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Durante o trabalho com erros de tempo de execução síncronos, use as instruções `try..catch..finally` para detectar erros. Quando ocorre um erro de tempo de execução, o tempo de execução do Flash lança uma exceção, o que significa que ele suspende a execução normal e cria um objeto especial do tipo Error. O objeto Error é lançado para o primeiro bloco `catch` disponível.

A instrução `try` delimita instruções com potencial para criar erros. A instrução `catch` é sempre usada com uma instrução `try`. Se for detectado um erro em uma das instruções no bloco `try`, as instruções `catch` anexadas a `try` serão executadas.

A instrução `finally` delimitará as instruções que serão executadas caso ocorra ou não um erro no bloco `try`. Se não houver nenhum erro, as instruções no bloco `finally` serão executadas depois que as instruções do bloco `try` forem concluídas. Se houver um erro, a instrução `catch` apropriada será executada primeiro, seguida pelas instruções no bloco `finally`.

O código a seguir demonstra a sintaxe para usar as instruções `try..catch..finally`:

```
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.
}
```

Cada instrução `catch` identifica um tipo específico de exceção que ela manipula. A instrução `catch` pode especificar apenas as classes de erro que forem subclasses da classe Error. Cada instrução `catch` é verificada por ordem. Somente a primeira instrução `catch` que corresponder ao tipo de erro gerado será executada. Em outras palavras, se você verificar primeiro a classe Error de alto nível e depois uma subclasse dela, somente a classe Error de alto nível será correspondente. O seguinte código ilustra essa questão:

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

O código anterior exibe a seguinte saída:

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

Para detectar corretamente o ArgumentError, os tipos de erro mais específicos devem ser listados primeiro e os mais genéricos por último, como mostra o seguinte código:

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

Vários métodos e propriedades na API do ActionScript lançam erros de tempo de execução quando encontram erros ao serem executados. Por exemplo, o método `close()` na classe Sound lançará um IOError se o método não conseguir fechar o fluxo de áudio, conforme demonstrado no seguinte código:

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

À medida que você se familiariza com a Referência do ActionScript® 3.0 para Adobe® Flash® Platform, irá perceber quais métodos geram exceções, como detalhado na descrição de cada método.

A instrução throw

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os tempos de execução do Flash lançam exceções quando encontram erros no seu aplicativo em tempo de execução. Além disso, você mesmo pode lançar exceções explicitamente usando a instrução `throw`. Ao lançar erros explicitamente, a Adobe recomenda que você lance ocorrências da classe Error ou de suas subclasses. O código a seguir demonstra uma instrução `throw` que lança uma ocorrência da classe Error, `MyErr` e finalmente chama uma função, `myFunction()`, para responder depois que o erro é lançado:

```
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.
}
```

Observe que as instruções `catch` são ordenadas de forma que os tipos de dados mais específicos sejam listados primeiro. Se a instrução `catch` do tipo de dados Number for listada primeiro, a instrução catch para o tipo de dados uint e a instrução catch para o tipo de dados int nunca serão executadas.

Nota: *Na linguagem de programação Java, cada função que pode lançar uma exceção deve declarar esse fato, listando as classes de exceção que pode lançar em uma cláusula* `throws` *anexada à declaração de função. O ActionScript não requer que você declare as exceções geradas por uma função.*

Exibição de uma mensagem de erro simples

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma das grandes vantagens do novo modelo de eventos de exceção e erro é que ele permite dizer aos usuários quando e por que uma ação falhou. Cabe a você escrever o código para exibir a mensagem e oferecer opções em resposta.

O código a seguir mostra uma instrução `try..catch` simples para exibir o erro em um campo de texto:

```
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);
            }
        }
    }
}
```

Usando uma ampla variedade de classes de erro e erros do compilador embutidos, o ActionScript 3.0 oferece mais informações do que as versões anteriores do ActionScript sobre o motivo de uma falha. Essa informação permite criar aplicativos mais estáveis com uma manipulação de erros melhor.

Relançamento de erros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao criar aplicativos, há várias ocasiões em que pode ser preciso relançar um erro quando não é possível manipulá-lo adequadamente. Por exemplo, o código a seguir mostra um bloco `try..catch` aninhado, que relança um ApplicationError personalizado quando o bloco `catch` não consegue manipular o erro:

```
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);
}
```

O resultado do snippet anterior seria o seguinte:

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

O bloco `try` aninhado lança um erro ApplicationError personalizado que é detectado pelo bloco `catch` subsequente. Esse bloco `catch` aninhado pode tentar manipular o erro e, se for bem-sucedido, lançará o objeto ApplicationError ao bloco `try..catch` delimitador.

Criação de classes de erros personalizadas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode estender uma das classes Error padrão para criar suas próprias classes de erro especializadas no ActionScript. Há vários motivos para criar suas próprias classes de erro:

- Para identificar erros ou grupos de erros específicos que são exclusivos do seu aplicativo.

  Por exemplo, execute ações diferentes para erros lançados por seu próprio código, além daqueles capturados pelo tempo de execução do Flash. Você pode criar uma subclasse da classe Error para controlar o novo tipo de dados de erro em blocos `try..catch`.

- Para fornecer recursos de exibição de erros exclusivos para os erros gerados pelo seu aplicativo.

  Por exemplo, você pode criar um novo método `toString()` que formata suas mensagens de erro de uma determinada maneira. Você também pode definir o método `lookupErrorString()` que obtém um código de erro e recupera a mensagem apropriada com base na preferência de idioma do usuário.

Uma classe de erro especializada deve estender a classe Error principal do ActionScript. Veja um exemplo de uma classe AppError especializada que estende a classe Error:

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

O exemplo a seguir mostra o uso de AppError em seu projeto:

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

Nota: *Para substituir o método `Error.toString()` na sua subclasse, atribua-lhe um parâmetro `...(rest)`. A especificação da linguagem ECMAScript, na qual o ActionScript 3.0 tem base, define o método `Error.toString()` desse modo, enquanto o ActionScript 3.0 adota a mesma definição para garantir a compatibilidade com versões anteriores. Portanto, ao substituir o método `Error.toString()`, deve haver uma correspondência exata de parâmetros. Os parâmetros não devem ser transmitidos para o método `toString()` em tempo de execução porque serão ignorados.*

Resposta a eventos e status de erros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um dos aprimoramentos mais notáveis da manipulação de erros no ActionScript 3.0 é o suporte à manipulação de eventos de erro para responder a erros assíncronos enquanto um aplicativo está em execução. (Para obter uma definição de erros assíncronos, consulte "Tipos de erros" na página 53.)

Você pode criar ouvintes e manipuladores de eventos para responder aos eventos de erro. Muitas classes despacham os eventos de erro da mesma forma o fazem com outros eventos. Por exemplo, uma instância da classe XMLSocket normalmente despacha três tipos de eventos: `Event.CLOSE`, `Event.CONNECT` e `DataEvent.DATA`. Entretanto, quando ocorre um problema, a classe XMLSocket pode despachar o `IOErrorEvent.IOError` ou o `SecurityErrorEvent.SECURITY_ERROR`. Para obter mais informações sobre ouvintes e manipuladores de eventos, consulte"Manipulação de eventos" na página 126.

Os eventos de erro se enquadram em duas categorias:

- Eventos de erro que estendem a classe ErrorEvent

  A classe flash.events.ErrorEvent contém as propriedades e os métodos para gerenciar erros relacionados às operações de rede e comunicação em um aplicativo em tempo de execução. As classes AsyncErrorEvent, IOErrorEvent e SecurityErrorEvent estendem a classe ErrorEvent. Se você estiver usando a versão de depurador de um tempo de execução do Flash, uma caixa de diálogo o informará, em tempo de execução, sobre qualquer evento de erro sem as funções de ouvinte que o player encontrar.

- Eventos de erro baseados no status

  Os eventos de erro baseados no status estão relacionados às propriedades `netStatus` e `status` das classes de rede e comunicação. Se o tempo de execução do Flash encontrar um problema ao ler ou gravar dados, o valor das propriedades `netStatus.info.level` ou `status.level` (dependendo do objeto de classe usado) será definido com o valor `"error"`. A resposta a esse erro é verificar se a propriedade `level` contém o valor `"error"` na função do manipulador de eventos.

Trabalho com eventos de erro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe ErrorEvent e suas subclasses contêm tipos de erro para manipular erros despachados pelos tempos de execução do Flash quando tentam ler ou gravar dados.

O exemplo a seguir usa uma instrução `try..catch` e manipuladores de evento de erro para exibir os erros detectados ao tentar ler um arquivo local. Você pode adicionar código de manipulação mais sofisticado para fornecer opções a um usuário ou manipular o erro automaticamente nos locais indicados pelo comentário "seu código de manipulação de erros aqui":

```
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
        }
    }
}
```

Trabalho com eventos de alteração de status

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os tempos de execução do Flash alteram dinamicamente o valor das propriedades `netStatus.info.level` ou `status.level` para as classes que suportam a propriedade `level` enquanto uma aplicativo está em execução. As classes que possuem a propriedade `netStatus.info.level` são: NetConnection, NetStream e SharedObject. As classes que possuem a propriedade `status.level` são: HTTPStatusEvent, Camera, Microphone e LocalConnection. Você pode escrever uma função de manipulador para responder à alteração no valor `level` e controlar os erros de comunicação.

O exemplo a seguir usa uma função `netStatusHandler()` para testar o valor da propriedade `level`. Se a propriedade `level` indicar que um erro foi encontrado, o código rastreia a mensagem “Video stream failed”.

```
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);
        }
    }
}
```

Comparação das classes Error

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript fornece várias classes Error predefinidas. Mas você também pode usar as mesmas classes Error no seu próprio código. Há dois tipos principais de classes Error no ActionScript 3.0: As classes Error principais do ActionScript e as classes Error do pacote flash.error. O pacote flash.error contém classes adicionais para auxiliar no desenvolvimento e na depuração de aplicativos do ActionScript 3.0.

classes Error principais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As classes de erro principais incluem as classes ArgumentError, EvalError, RangeError, ReferenceError, SecurityError, SyntaxError, TypeError, URIError e VerifyError. Cada uma dessas classes está localizada no namespace de nível superior.

Nome da classe	Descrição	Observações
Error	A classe Error é usada para gerar exceções e é a classe base para as outras classes de exceção definidas no ECMAScript: EvalError, RangeError, ReferenceError, SyntaxError, TypeError e URIError.	A classe Error serve de classe base para todos os erros de tempo de execução e é a classe base recomendada para qualquer classe de erro personalizada.
ArgumentError	A classe ArgumentError representa um erro que ocorre quando os valores de parâmetro fornecidos durante uma chamada de função não correspondem aos definidos para essa função.	Alguns exemplos de erros de argumento incluem: • Foram fornecidos argumentos insuficientes ou excedentes para um método. • Um argumento que deveria ser um membro de uma enumeração não é.
EvalError	Uma exceção EvalError é lançada se qualquer parâmetro for transmitido para o construtor da classe Function ou se o código do usuário chamar a função `eval()`.	No ActionScript 3.0, o suporte à função `eval()` foi removido e as tentativas de usar a função resultam em erro. As versões anteriores do Flash Player usavam a função `eval()` para acessar variáveis, propriedades, objetos ou clipes de filme pelo nome.
RangeError	Uma exceção RangeError é lançada quando um valor numérico fica fora de uma faixa aceitável.	Por exemplo, um RangeError é lançado pela classe Timer se um atraso for negativo ou se não for finito. Um RangeError também poderia ser lançado se você tentasse adicionar um objeto de exibição em uma profundidade inválida.
ReferenceError	Uma exceção de ReferenceError é lançada quando há uma tentativa de referência a uma propriedade não definida em um objeto selado (não dinâmico). As versões de compilador do ActionScript antes do ActionScript 3.0 não lançavam um erro quando havia uma tentativa de acesso a uma propriedade `não definida`. No entanto, o ActionScript 3.0 lança a exceção ReferenceError nessa condição.	As exceções para variáveis não definidas apontam para possíveis bugs, ajudando a melhorar a qualidade do software. Entretanto, se você não está acostumado a inicializar as variáveis, esse novo comportamento do ActionScript exige algumas alterações nos seus hábitos de criação de código.

Nome da classe	Descrição	Observações
SecurityError	A exceção SecurityError é lançada quando ocorre uma violação de segurança e o acesso é negado.	Alguns exemplos de erros de segurança incluem: • Um acesso a propriedade ou uma chamada de método não autorizado é feito no outro lado de um limite de caixa de proteção. • Foi feita uma tentativa de acessar uma URL não permitida pela caixa de proteção. • Foi feita uma tentativa de conexão de soquete a uma porta, mas o arquivo de política de soquete necessário não estava disponível. • Foi feita uma tentativa de acessar a câmera ou o microfone do usuário, e o usuário negou o acesso ao dispositivo.
SyntaxError	Uma exceção SyntaxError é lançada quando ocorre um erro de análise no código do ActionScript.	Um SyntaxError pode ser lançado sob as seguintes circunstâncias: • O ActionScript lança exceções SyntaxError quando a classe RegExp desmembra uma expressão regular inválida. • O ActionScript lança exceções SyntaxError quando uma classe XMLDocument desmembra um XML inválido.
TypeError	A exceção TypeError é lançada quando o tipo real de operando é diferente do tipo esperado.	Um TypeError pode ser lançado sob as seguintes circunstâncias: • Um parâmetro real para uma função ou um método não pôde ser forçado ao tipo de parâmetro formal. • Um valor é atribuído a uma variável e não pode ser forçado ao tipo da variável. • O lado direito do operador `is` ou `instanceof` não é um tipo válido. • A palavra-chave `super` é usada ilegalmente. • Uma pesquisa de propriedades resulta em mais de uma ligação e, portanto, é ambígua. • Um método é chamado em um objeto incompatível. Por exemplo, uma exceção TypeError será lançada se um método na classe RegExp for "enxertado" em um objeto genérico e, depois, chamado.
URIError	Uma exceção URIError é lançada quando uma das funções de manipulação de URI global é usada de forma incompatível com sua definição.	Um URIError pode ser lançado sob as seguintes circunstâncias: Um URI inválido é especificado para uma função da API do Flash Player que espera um URI válido, como `Socket.connect()`.
VerifyError	Uma exceção VerifyError é lançada quando é encontrado um arquivo SWF malformado ou corrompido.	Quando um arquivo SWF carrega outro arquivo SWF, o SWF pai pode detectar um VerifyError gerado pelo SWF carregado.

flash.error, classes Error do pacote

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O pacote flash.error contém classes de Error que são consideradas parte da API do tempo de execução do Flash. Ao contrário das classes Error descritas, o pacote flash.error comunica eventos de erro específicos dos tempos de execução do Flash (como Flash Player e Adobe AIR).

Nome da classe	Descrição	Observações
EOFError	Uma exceção EOFError é lançada quando você tenta ler além do fim dos dados disponíveis.	Por exemplo, um EOFError é mostrado quando um dos métodos de leitura na interface IDataInput é chamado e não há dados suficientes para atender à solicitação de leitura.
IllegalOperationError	Uma exceção IllegalOperationError é lançada quando um método não é implementado ou quando a implementação não abrange o uso atual.	Exemplos de exceções de erro de operação ilegal incluem: • Uma classe base, como DisplayObjectContainer, fornece mais funcionalidade do que o Palco pode suportar. Por exemplo, se você tentar obter ou definir uma máscara em Stage (usando `stage.mask`), o tempo de execução do Flash lança um IllegalOperationError com a mensagem: "A classe Stage não implementa essa propriedade ou esse método". • Uma subclasse herda um método que não requer e para o qual não deseja oferecer suporte. • Determinados métodos de acessibilidade são chamados quando o Flash Player é compilado sem o suporte de acessibilidade. • Recursos somente de autoria são chamados de uma versão de tempo de execução do Flash Player. • Você tenta definir o nome de um objeto colocado na linha de tempo.
IOError	Uma exceção IOError é lançada quando ocorre algum tipo de exceção de E/S.	Por exemplo, você obtém este erro quando uma operação de leitura/gravação é tentada em um soquete que não está conectado ou se desconectou.
MemoryError	Uma exceção MemoryError é lançada quando há uma falha na solicitação de alocação de memória.	Por padrão, a ActionScript Virtual Machine 2 não impõe um limite sobre a quantidade de memória que um programa do ActionScript aloca. Em um computador de mesa, as falhas de alocação de memória não são frequentes. Um erro é lançado quando o sistema não consegue alocar a memória necessária para uma operação. Por isso, em um computador de mesa, essa exceção é rara, a menos que uma solicitação de alocação seja extremamente grande; por exemplo, uma solicitação de 3 bilhões de bytes é impossível porque um programa Microsoft® Windows® de 32 bits só pode acessar 2 GB de espaço de endereçamento.
ScriptTimeoutError	Uma exceção ScriptTimeoutError é lançada quando um intervalo de tempo limite do script de 15 segundos é atingido. Ao detectar uma exceção ScriptTimeoutError, é possível manipular melhor o tempo limite do script. Se não houver um manipulador de exceções, a exceção não capturada exibirá uma caixa de diálogo com uma mensagem de erro.	Para impedir que um desenvolvedor mal-intencionado detecte a exceção e permaneça em um loop infinito, somente a primeira exceção ScriptTimeoutError lançada durante um determinado script pode ser detectada. Uma exceção ScriptTimeoutError subsequente não poderá ser detectada pelo seu código e irá imediatamente para o manipulador de exceções não detectadas.
StackOverflowError	A exceção StackOverflowError é lançada quando a pilha disponível para o script é esgotada.	Uma exceção StackOverflowError pode indicar que ocorreu recursão infinita.

Exemplo de tratamento de erros: aplicativo CustomErrors

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo CustomErrors demonstra técnicas para trabalhar com erros personalizados ao criar um aplicativo. Estas técnicas são:

- Validar um pacote XML
- Escrever um erro personalizado
- Lançar erros personalizados
- Notificar os usuários quando um erro é lançado

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo CustomErrors podem ser encontrados na pasta Samples/CustomError. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
CustomErrors.mxml ou CustomErrors.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML)
com/example/programmingas3/errors/ApplicationError.as	Uma classe que serve de classe de erro base para as classes FatalError e WarningError.
com/example/programmingas3/errors/FatalError.as	Uma classe que define um erro FatalError lançado pelo aplicativo. Essa classe estende a classe ApplicationError personalizada.
com/example/programmingas3/errors/Validator.as	Uma classe que define um único método que valida um pacote XML funcionário fornecido pelo usuário.
com/example/programmingas3/errors/WarningError.as	Uma classe que define um erro WarningError lançado pelo aplicativo. Essa classe estende a classe ApplicationError personalizada.

Visão geral do aplicativo CustomErrors

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o aplicativo é carregado, o método `initApp()` é chamado para os aplicativos Flex ou o código (não função) de linha de tempo é executado para os aplicativos do Flash Professional. Esse código define um pacote XML de exemplo a ser verificado pela classe Validator. O seguinte código é executado:

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

O pacote XML é exibido posteriormente em uma ocorrência do componente TextArea no Palco. Esta etapa permite modificar o pacote XML antes de tentar validá-lo novamente.

Quando o usuário clica no botão Validate, o método `validateData()` é chamado. Esse método valida o pacote XML funcionário usando o método `validateEmployeeXML()` na classe Validator. O código a seguir mostra o método `validateData()`:

```
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);
    }
}
```

Primeiro, um objeto XML temporário é criado usando o conteúdo da ocorrência do componente TextArea `xmlText`. Em seguida, o método `validateEmployeeXML()` na classe Validator personalizada (com.example.programmingas3/errors/Validator.as) é chamado e transmite o objeto XML temporário como um parâmetro. Se o pacote XML for válido, a ocorrência do componente Label `status` exibirá uma mensagem de êxito e o aplicativo será encerrado. Se o método `validateEmployeeXML()` lançar um erro personalizado (isto é, se ocorrer um FatalError, WarningError ou um Error genérico), a instrução `catch` apropriada executará e chamará o método `showFatalError()`, `showWarningError()` ou `showGenericError()`. Cada um desses métodos exibe uma mensagem apropriada em uma área de texto chamada `statusText` para notificar o usuário do erro específico que ocorreu. Cada método também atualiza a ocorrência do componente Label `status` com uma mensagem específica.

Se ocorrer um erro fatal durante a tentativa de validar o pacote XML funcionário, a mensagem de erro será exibida na área de texto `statusText` e as ocorrências dos componentes TextArea `xmlText` e Button `validateBtn` serão desativadas, como mostra o seguinte código:

```
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();
}
```

Se ocorrer um erro de aviso, em vez de um erro fatal, a mensagem de erro será exibida na ocorrência `statusText` TextArea, mas as ocorrências dos componentes Button e TextField `xmlText` não serão desativadas. O método `showWarningError()` exibe uma mensagem de erro personalizada na área de texto `statusText`. A mensagem também solicita que o usuário decida se deseja prosseguir com a validação do XML ou cancelar o script. O seguinte trecho mostra o método `showWarningError()`:

```
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;
}
```

Quando o usuário clica no botão Yes ou No, o método `closeHandler()` é chamado. O seguinte trecho mostra o método `closeHandler()`:

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

Se o usuário decidir cancelar o script clicando em Yes, um FatalError será lançado fazendo com que o aplicativo seja encerrado.

Criação de um validador personalizado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Validator personalizada contém um único método, `validateEmployeeXML()`. O método `validateEmployeeXML()` usa um único argumento, `employee`, que é o pacote XML que você deseja validar. O método `validateEmployeeXML()` é o seguinte:

```
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);
    }
}
```

Para ser validado, um funcionário deve pertencer a um (e apenas um) centro de custos. Se o funcionário não pertencer a nenhum centro de custos, o método lançará um FatalError, que é emitido ao método `validateData()` no arquivo do aplicativo principal. Se o funcionário pertencer a mais de um centro de custos, será lançado um WarningError. A verificação final no validador XML é que o usuário tem exatamente um número de seguro social (o nó `ssn` no pacote XML). Se não houver exatamente um nó `ssn`, será lançado um erro FatalError.

Você pode adicionar outras verificações ao método `validateEmployeeXML()`, por exemplo, para garantir que o nó `ssn` contenha um número válido ou que o funcionário tenha pelo menos um número de telefone e um endereço de email definido, e os dois valores sejam válidos. Também é possível modificar o XML de forma que cada funcionário tenha uma ID exclusiva e especifique a ID do seu gerente.

Definição da classe ApplicationError

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe ApplicationError serve de classe base para as classes FatalError e WarningError. A classe ApplicationError estende a classe Error e define seus próprios métodos e propriedades personalizados, incluindo a definição de ID e gravidade de um erro, e um objeto XML que contém os códigos e as mensagens de erro personalizados. Essa classe também define duas constantes estáticas que são usadas para definir a gravidade de cada tipo de erro.

O método do construtor da classe ApplicationError é o seguinte:

```
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>;
}
```

Cada nó de erro no objeto XML contém um código numérico exclusivo e uma mensagem de erro. As mensagens de erro podem ser facilmente pesquisadas pelo seu código de erro usando o E4X, como mostra o seguinte método `getMessageText()`:

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

O método `getMessageText()` usa um único argumento inteiro, `id`, e retorna uma sequência de caracteres. O argumento `id` é o código de erro para o erro a ser pesquisado. Por exemplo, transmitir um `id` de 9001 recupera o erro dizendo que os funcionários devem ser atribuídos apenas a um centro de custos. Se houver mais de um erro com o mesmo código, o ActionScript retornará a mensagem de erro apenas para o primeiro resultado encontrado (`message[0]` no objeto XMLList retornado).

O método seguinte nessa classe, `getTitle()`, não usa nenhum parâmetro e retorna um valor de sequência de caracteres que contém a ID de erro para esse erro específico. Esse valor é usado para ajudar a identificar com facilidade o erro exato que ocorreu durante a validação do pacote XML. O seguinte trecho mostra o método `getTitle()`:

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

O método final na classe ApplicationError é `toString()`. Esse método substitui a função definida na classe Error para que você possa personalizar a apresentação da mensagem de erro. O método retorna uma sequência de caracteres que identifica o número do erro específico e a mensagem que ocorreu.

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

Definição da classe FatalError

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe FatalError estende a classe ApplicationError personalizada e define três métodos: o construtor FatalError, `getTitle()` e `toString()`. O primeiro método, o construtor FatalError, usa um único argumento inteiro, `errorID`, define a gravidade do erro usando os valores constantes estáticos definidos na classe ApplicationError e obtém a mensagem de erro específica, chamando o método `getMessageText()` na classe ApplicationError. O construtor FatalError é o seguinte:

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

O método seguinte na classe FatalError, `getTitle()`, substitui o método `getTitle()` definido anteriormente na classe ApplicationError e anexa o texto "-- FATAL" no título para informar ao usuário que um erro fatal ocorreu. O método `getTitle()` é o seguinte:

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

O método final nessa classe, `toString()`, substitui o método `toString()` definido na classe ApplicationError. O método `toString()` é:

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

Definição da classe WarningError

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe WarningError estende a classe ApplicationError e é quase idêntica à classe FatalError, exceto por duas pequenas alterações de sequência de caracteres e por definir a gravidade do erro como ApplicationError.WARNING, em vez de ApplicationError.FATAL, como mostra o seguinte código:

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

Capítulo 5: Uso de expressões regulares

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma expressão regular descreve um padrão que é utilizado para localizar e manipular texto correspondente em strings. As expressões regulares parecem ser strings, mas elas podem incluir códigos especiais para descrever padrões e repetição. Por exemplo, a expressão regular a seguir corresponde a uma string que começa com o caractere A seguido por um ou mais dígitos sequenciais.

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

Os tópicos a seguir descrevem a sintaxe básica para construir expressões regulares. Entretanto, as expressões regulares podem ter muita complexidade e nuances. Você pode encontrar informações detalhadas sobre expressões regulares na Web e nas livrarias. Lembre-se de que diferentes ambientes de programação implementam expressões regulares de modos diferentes. O ActionScript 3.0 implementa expressões regulares como definido na especificação de idioma do ECMAScript Edição 3 (ECMA-262).

Mais tópicos da Ajuda

RegExp

Noções básicas de expressões regulares:

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma expressão regular descreve um padrão de caracteres. As expressões regulares são normalmente usadas para verificar se um valor de texto está em conformidade com um determinado padrão (como a verificação para saber se o número de telefone digitado pelo usuário tem o número de dígitos adequado) ou para substituir partes de um valor de texto que corresponde a um determinado padrão.

As expressões regulares podem ser simples. Por exemplo, suponha que você queira confirmar se uma determinada string corresponde a "ABC" ou substituir cada ocorrência de "ABC" em uma string por algum outro texto. Nesse caso, você pode usar a seguinte expressão regular, que define o padrão composto pelas letras A, B e C em sequência:

```
/ABC/
```

Observe que o literal de uma expressão regular é delineado com o caractere de barra (`/`).

Os padrões de expressões regulares também podem ser complexos e, às vezes, críptico na aparência, como a expressão a seguir, para corresponder a uma endereço de email válido:

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

Com mais frequência, você utilizará expressões regulares para pesquisar padrões em strings e substituir caracteres. Nesses casos, você criará um objeto de expressão regular e o usará como um parâmetro para um dos vários métodos da classe String. Os seguintes métodos da classe String usam as expressões regulares como parâmetros: `match()`, `replace()`, `search()` e `split()`. Para obter mais informações sobre esses métodos, consulte "Localização de padrões em sequências de caracteres e substituição de subsequências de caracteres" na página 17.

A classe RegExp inclui as seguintes opções: `test()` e `exec()`. Para obter mais informações, consulte "Métodos para usar expressões regulares com strings" na página 90.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que são relevantes a este recurso:

Caractere Escape Um caractere indicando que o caractere seguinte deve ser tratado como um metacaractere em vez de um caractere literal. Na sintaxe de expressões regulares, o caractere de barra invertida (\) é o caractere escape; portanto uma barra invertida seguida por outro caractere é um código especial e não apenas o próprio caractere.

Sinalizador Um caractere que especifica algumas opções sobre como o padrão de expressão regular deve ser utilizado, como distinguir entre caracteres maiúsculos e minúsculos.

Metacaractere Um caractere que tem um significado especial em um padrão de expressão regular, em oposição à representação literal do caractere no padrão.

Quantificador Um caractere (ou vários) indicando quantas vezes uma parte do padrão deve se repetir. Por exemplo, um quantificador é utilizado para designar que o código postal dos Estados Unidos deve conter cinco ou nove números.

Expressão regular Uma instrução do programa que define um padrão de caracteres que podem ser usados para confirmar se outras strings correspondem àquele padrão ou substituir partes de uma string.

Sintaxe da expressão regular

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Esta seção descreve todos os elementos da sintaxe de expressão regular do ActionScript. Como você verá, as expressões regulares podem ter muita complexidade e nuances. Você pode encontrar informações detalhadas sobre expressões regulares na Web e nas livrarias. Lembre-se de que diferentes ambientes de programação implementam expressões regulares de modos diferentes. O ActionScript 3.0 implementa expressões regulares como definido na especificação de idioma do ECMAScript Edição 3 (ECMA-262).

Normalmente, você usa expressões regulares que correspondem a padrões mais complicados do que uma string de caracteres simples. Por exemplo, a seguinte expressão regular define o padrão composto pelas letras A, B e C em sequência seguida por qualquer dígito:

```
/ABC\d/
```

O código `\d` representa “qualquer dígito”. O caractere de barra invertida (`\`) é chamado de caractere escape e é combinado ao caractere que o segue (nesse caso a letra d), tendo um significado especial na expressão regular.

A expressão regular a seguir define o padrão das letras ABC seguido por qualquer número de dígitos (observe o asterisco):

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

O caractere asterisco (`*`) é um *metacaractere*. Um metacaractere é um caractere que tem significado especial nas expressões regulares. O asterisco é um tipo específico de metacaractere chamado de *quantificador* que é usado para quantificar o número de repetição de um caractere ou de um grupo de caracteres. Para obter mais informações, consulte “Quantificadores” na página 82.

Além desse padrão, uma expressão regular pode conter sinalizadores, que especificam como ela deve ser correspondida. Por exemplo, a seguinte expressão regular usa o sinalizador `i`, que especifica que a expressão regular não diferencia maiúsculas de minúsculas na correspondência de strings:

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

Para obter mais informações, consulte “Sinalizadores e propriedades” na página 87.

Você pode usar expressões regulares com os seguintes métodos da classe String: `match()`, `replace()` e `search()`. Para obter mais informações sobre esses métodos, consulte "Localização de padrões em sequências de caracteres e substituição de subsequências de caracteres" na página 17.

Criação de uma ocorrência de uma expressão regular

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Há duas maneiras de criar uma ocorrência de expressão regular: Um modo usa caracteres de barra (`/`) para delinear a expressão regular; o outro usa o construtor `new`. Por exemplo, as expressões regulares a seguir são equivalentes:

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

As barras delineiam um literal da expressão regular da mesma forma que as aspas delineiam um literal de string. A parte da expressão regular entre as barras define o *padrão.* A expressão regular também pode incluir *sinalizadores* depois da barra delineadora final. Esses sinalizadores são considerados como parte da expressão regular, mas são separados do padrão.

Ao usar o construtor `new`, você usa duas strings para definir a expressão regular. A primeira string define o padrão e a segunda string define os sinalizadores como no exemplo a seguir:

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

Ao incluir uma barra *em* uma expressão regular que é definida utilizando os delineadores de barra, você deve preceder a barra com um caractere escape de barra invertida (`\`). Por exemplo, as expressões regulares a seguir correspondem ao padrão `1/2`:

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

Para incluir aspas *em* uma expressão regular que é definida com o construtor `new`, você deve adicionar uma caractere escape de barra invertida (`\`) antes das aspas (assim como faria ao definir qualquer literal String). Por exemplo, as expressões regulares a seguir correspondem ao padrão `eat at "joe's"`:

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

Não use o caractere escape de barra invertida com aspas em expressões regulares que são definidas utilizando os delineadores de barra. Da mesma forma, não use o caractere escape com barras em expressões regulares que são definidas utilizando o construtor `new`. As expressões regulares a seguir são equivalentes e definem o padrão `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"', '');
```

Em uma expressão regular definida com o construtor `new`, para usar uma metassequência que comece com o caractere de barra invertida (`\`) , como `\d` (que corresponde a qualquer dígito), digite duas vezes esse caractere de barra invertida:

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

É necessário digitar o caractere de barra invertida duas vezes nesse caso, pois o primeiro parâmetro do método do construtor `RegExp()` é uma string, e, em um literal de string, é necessário digitar um caractere de barra invertida duas vezes para que ele seja reconhecido como um único caractere de barra invertida.

As seções a seguir descrevem a sintaxe para definir os padrões de expressões regulares.

Para obter mais informações sobre sinalizadores, consulte "Sinalizadores e propriedades" na página 87.

Caracteres, metacaracteres e metassequências

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A expressão regular mais simples é aquela que corresponde a uma sequência de caracteres, como no exemplo a seguir:

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

Entretanto, os seguintes caracteres, conhecidos como metacaracteres , têm significado especial nas expressões regulares:

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

Por exemplo, a expressão regular a seguir corresponde à letra A seguida por nenhuma ou mais ocorrências da letra B (o metacaractere asterisco indica essa repetição), seguida pela letra C:

```
/AB*C/
```

Para incluir um metacaractere sem seu significado especial em um padrão de expressão regular, você deve usar o caractere escape de barra invertida (`\`). Por exemplo, a seguinte expressão regular corresponde à letra A seguida pela letra B, seguida por um asterisco e pela letra C:

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

Uma *metassequência*, como um metacaractere, tem um significado especial em uma expressão regular. Uma metassequência é composta por mais de um caractere. As seções a seguir fornecem detalhes sobre o uso de metacaracteres e metassequências.

Sobre metacaracteres

A tabela a seguir resume os metacaracteres que você pode usar em expressões regulares:

Metacaractere	Descrição
^ (circunflexo)	Correspondente ao início da string. Com o sinalizador `m` (`multiline`) definido, o circunflexo também corresponde ao início de uma linha (consulte "Sinalizadores e propriedades" na página 87). Observe que, quando utilizado no início de uma classe de caracteres, o circunflexo indica negação e não o início de uma string. Para obter mais informações, consulte "Classes de caracteres" na página 81.
$(sinal de dólar)	Correspondente ao fim da string. Com o conjunto de sinalizadores `m` (`multiline`), `$` também corresponde à posição antes de um caractere de nova linha (`\n`). Para obter mais informações, consulte "Sinalizadores e propriedades" na página 87.
(barra invertida) \	Elimina o significado do metacaractere especial dos caracteres especiais. Além disso, use o caractere de barra invertida se você quiser utilizar o caractere de barra em um literal de expressão regular, como em `/1\/2/` (para corresponder ao caractere 1, seguido pelo caractere de barra e pelo caractere 2).
. (ponto)	Corresponde a qualquer caractere único. Um ponto corresponde a um caractere de nova linha (`\n`) apenas se o sinalizador `s` (`dotall`) está definido. Para obter mais informações, consulte "Sinalizadores e propriedades" na página 87.
* (estrela)	Corresponde ao item anterior repetido nenhuma ou várias vezes. Para obter mais informações, consulte "Quantificadores" na página 82.
+ (adição)	Corresponde ao item anterior repetido uma ou várias vezes. Para obter mais informações, consulte "Quantificadores" na página 82.
? (ponto de interrogação)	Corresponde ao item anterior repetido nenhuma ou uma vezes. Para obter mais informações, consulte "Quantificadores" na página 82.

Metacaractere	Descrição
`(` e `)`	Define grupos dentro de uma expressão regular. Use os grupos para: • Confinar o escopo do alternador |: `/(a|b|c)d/` • Definir o escopo de um quantificador: `/(walla.){1,2}/` • Em referências anteriores: Por exemplo, `\1` na expressão regular a seguir corresponde àquilo que correspondeu ao primeiro grupo entre parênteses do padrão: • `/(\w*) é repetido: \1/` Para obter mais informações, consulte "Grupos" na página 84.
`[` e `]`	Define uma classe de caracteres que define possíveis correspondências para um único caractere: `/[aeiou]/` corresponde a qualquer um dos caracteres especificados. Nas classes de caracteres, use o hífen (`-`) para designar um intervalo de caracteres: `/[A-Z0-9]/` corresponde às letras maiúsculas de A a Z ou de 0 a 9. Nas classes de caracteres, insira uma barra invertida para eliminar os caracteres] e -: `/[+\-]\d+/` corresponde a `+` ou `-` antes de um ou mais dígitos. Nas classes de caracteres, outros caracteres - normalmente metacaracteres -, não tratados como caracteres normais (e não metacaracteres), sem a necessidade de usar uma barra invertida: `/[$]/£` corresponde a `$` ou £. Para obter mais informações, consulte "Classes de caracteres" na página 81.
| *(pipe)*	Utilizado para alternação, para corresponder à parte do lado esquerdo ou do lado direito: `/abc|xyz/` corresponde a `abc` ou `xyz`.

Sobre metassequências

As metassequências são sequências de caracteres que têm significado especial em um padrão de expressão regular. A tabela a seguir descreve essas metassequências:

Metassequência	Descrição

`{`***n***`}`
`{`***n***`,}`
e
`{`***n***`,`***n***`}`
	Especifica um quantificador numérico ou intervalo de quantificador para o item anterior:
	`/A{27}/` corresponde ao caractere `A` repetido `27` vezes.
	`/A{3,}/` corresponde ao caractere `A` repetido `3` vezes.
	`/A{3,5}/` corresponde ao caractere `A` repetido de `3` a `5` vezes.
	Para obter mais informações, consulte "Quantificadores" na página 82.

`\b`	Corresponde à posição entre um caractere de palavra e um diferente de palavra. Se o primeiro ou o último caractere na string for um caractere de palavra, ele também corresponderá ao início ou ao fim da string.
`\B`	Corresponde à posição entre dois caracteres de palavra. Também corresponde à posição entre dois caracteres diferentes de palavra.
`\d`	Corresponde a um dígito decimal.
`\D`	Corresponde a qualquer caractere diferente de dígito.
`\f`	Corresponde ao caractere feed de formulário.
`\n`	Corresponde ao caractere de nova linha.

Metassequência	Descrição
`\r`	Corresponde ao caractere de retorno.
`\s`	Corresponde a qualquer caractere de espaço em branco (um caractere de espaço, tabulação, nova linha ou retorno).
`\S`	Corresponde a qualquer caractere diferente de um caractere de espaço em branco.
`\t`	Corresponde ao caractere de tabulação.
`\u`***nnnn***	Corresponde ao caractere Unicode com o código de caractere especificado pelo número hexadecimal *nn*. Por exemplo, `\u263a` é um caractere smiley.
`\v`	Corresponde ao caractere feed vertical.
`\w`	Corresponde a um caractere de palavra (`AZ`–, `az`–, `0-9` ou_). Observe que `\w` não corresponde a caracteres diferentes de inglês, como `é` ,`ñ` ou`ç` .
`\W`	Corresponde a qualquer caractere diferente de um caractere de palavra.
`\\x`***nn***	Corresponde ao caractere com o valor ASCII especificado, como definido pelo número hexadecimal *nn*.

Classes de caracteres

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você usa as classes de caracteres para especificar uma lista de caracteres que correspondem a uma posição na expressão regular. Você define as classes de caracteres com colchetes (`[` e `]`). Por exemplo, a expressão regular a seguir define uma classe de caracteres que corresponde a `bag`, `beg`, `big`, `bog` ou`bug`:

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

Sequências de eliminação nas classes de caracteres

A maioria dos metacaracteres e das metassequências que normalmente tem significados especiais em uma expressão regular*não* tem esses mesmos significados em uma classe de caractere. Por exemplo, em uma expressão regular, o asterisco é usado para repetição, mas esse não é o caso quando o asterisco aparece em uma classe de caracteres. A classe de caracteres a seguir corresponde ao asterisco literalmente, junto com qualquer outro caractere listado:

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

Entretanto, os três caracteres listados na tabela a seguir funcionam como metacaracteres, com significado especial, nas classes de caracteres:

Metacaractere	Significado nas classes de caracteres
`]`	Define o final de uma classe de caracteres.
`-`	Define um intervalo de caracteres (consulte a próxima seção, “Intervalos de caracteres nas classes de caracteres”).
`\`	Define metassequências e elimina o significação especial dos metacaracteres.

Para que qualquer um desses caracteres seja reconhecido como caracteres literais (sem o significado do metacaractere especial), você deve precedê-lo com o caractere escape de barra invertida. Por exemplo, a expressão regular a seguir inclui uma classe de caracteres que corresponde a qualquer um dos quatro símbolos (`$`, `\`, `]` ou`-`):

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

Além dos metacaracteres que mantêm seu significado especial, as metassequências a seguir funcionam como metassequências nas classes de caracteres:

Metassequência	Significado nas classes de caracteres
`\n`	Corresponde a um caractere de nova linha.
`\r`	Corresponde a um caractere de retorno.
`\t`	Corresponde a um caractere de tabulação.
`\u`***nnnn***	Corresponde ao caractere com o valor do ponto de código Unicode especificado (como definido pelo número hexadecimal *nnnn*).
`\\x`***nn***	Corresponde ao caractere com o valor ASCII especificado (como definido pelo número hexadecimal *nn*).

Outros metacaracteres e metassequências de expressões regulares são tratados como caracteres normais em uma classe de caracteres.

Intervalos de caracteres nas classes de caracteres

Use o hífen para especificar um intervalo de caracteres, como `A-Z`, `a-z` ou`0-9`. Esses caracteres devem constituir um intervalo válido no conjunto de caracteres. Por exemplo, a classe de caracteres a seguir corresponde a qualquer um dos caracteres no intervalo de `a-z` ou qualquer dígito:

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

Você também pode usar o código de caractere ASCII `\\x`*nn* para especificar um intervalo por valor ASCII. Por exemplo, a classe de caracteres a seguir corresponde a qualquer caractere de um conjunto de caracteres ASCII estendido (como `é` e `ê`):

```
\\x
```

Classes de caracteres negadas

Quando você usa um caractere circunflexo (`^`) no início de uma classe de caracteres, ele nega aquela classe — qualquer caractere não listado é considerado uma correspondência. A classe de caracteres a seguir corresponde a qualquer caractere *exceto* a uma letra minúscula (`az–`) ou um dígito:

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

Você deve digitar o caractere circunflexo (`^`) no *início* de uma classe de caracteres para indicar a negação. Caso contrário, você estará simplesmente adicionando o caractere circunflexo aos caracteres na classe de caracteres. Por exemplo, a classe de caracteres a seguir corresponde a qualquer um dos caracteres de símbolo, incluindo o circunflexo:

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

Quantificadores

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você usa quantificadores para especificar repetições de caracteres ou sequências nos padrões, como se segue:

Metacaractere de quantificador	Descrição
* (estrela)	Corresponde ao item anterior repetido nenhuma ou várias vezes.
+ (adição)	Corresponde ao item anterior repetido uma ou várias vezes.
? (ponto de interrogação)	Corresponde ao item anterior repetido nenhuma ou uma vezes.
{*n*} {*n*,} e {*n*,*n*}	Especifica um quantificador numérico ou intervalo de quantificador para o item anterior: /A{27}/ corresponde ao caractere A repetido 27 vezes. /A{3,}/ corresponde ao caractere A repetido 3 ou mais vezes. /A{3,5}/ corresponde ao caractere A repetido de 3 a 5 vezes.

Você pode aplicar um quantificador a um único caractere, a uma classe de caracteres ou a um grupo:

- /a+/ corresponde ao caractere a repetido uma ou mais vezes.
- /\d+/ corresponde a um ou mais dígitos.
- /[abc]+/ corresponde a uma repetição de um ou mais caracteres, cada um é a, b ou c.
- /(very,)*/ corresponde à palavra very seguida por uma vírgula e um espaço repetido nenhuma ou várias vezes.

Você pode usar quantificadores em grupos entre parênteses que têm quantificadores aplicados a eles. Por exemplo, o quantificador a seguir corresponde a strings como word e word-word-word:

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

Por padrão, as expressões regulares executam o que é conhecido como correspondência *greedy*. Qualquer subpadrão na expressão regular (como .*) tenta corresponder o máximo possível de caracteres na string antes de avançar para a próxima parte da expressão regular. Por exemplo, considere a seguinte expressão regular e string:

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

A expressão regular corresponde à string inteira:

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

Considere, entretanto, que você deseja corresponder apenas um grupo ... . É possível fazer isso da seguinte forma:

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

Adicione um ponto de interrogação (?) depois de qualquer quantificador para alterá-lo para o que conhecido como *quantificador lazy*. Por exemplo, a seguinte expressão regular, que usa o quantificador lazy *?, corresponde a seguido pelo número mínimo de caracteres possíveis (lazy) e por :

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

Lembre os seguintes pontos sobre quantificadores:

- Os quantificadores {0} e {0,0} não excluem um item de uma correspondência.
- Não combine vários quantificadores, como em /abc+*/.
- O dot (.) não estende linhas a menos que o sinalizador s (dotall) esteja definido, mesmo se for seguido por um quantificador *. Por exemplo, considere o seguinte código:

```
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>
```

Para obter mais informações, consulte "Sinalizadores e propriedades" na página 87.

Alternação

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use o caractere `|` (pipe) em uma expressão regular para que o mecanismo de expressões regulares considere as alternativas para uma correspondência. Por exemplo, as expressões regulares a seguir correspondem a qualquer uma das palavras `cat, dog, pig, rat`:

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

Você pode usar parênteses para definir grupos a fim de restringir o escopo do alternador |. A expressão regular a seguir corresponde a `cat` seguido por `nap` ou`nip`:

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

Para obter mais informações, consulte "Grupos" na página 84.

As duas expressões regulares a seguir, uma usando o alternador `|` e a outra usando uma classe de caracteres (definida com `[` e `]`), são equivalentes:

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

Para obter mais informações, consulte "Classes de caracteres" na página 81.

Grupos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode especificar um grupo em uma expressão regular utilizando parênteses, como se segue:

```
/class-(\d*)/
```

Um grupo é uma subseção de um padrão. Você pode usar grupos para fazer as seguintes atividades:

- Aplicar um quantificador a mais de um caractere.
- Delinear subpadrões a serem aplicados com alternação (utilizando o caractere `|`).
- Capturar correspondências de substring (por exemplo, utilizando `\1` em uma expressão regular para corresponder um grupo com correspondência anterior ou utilizando `$1` de modo semelhante no método `replace()` da classe String).

As seções a seguir fornecem detalhes sobre o uso de grupos.

Uso de grupos com quantificadores

Se você não usar um grupo, um quantificador se aplicará ao caractere ou classe de caracteres que o precede, como mostrado a seguir:

```
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
```

Entretanto, você pode usar um grupo para aplicar um quantificador a mais de um caractere ou classe de caracteres:

```
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
```

Para obter mais informações sobre quantificadores, consulte “Quantificadores” na página 82.

Uso de grupos com o caractere alternador (|)

Você pode usar grupos para definir o grupo de caracteres aos quais deseja aplicar um caractere alternador (`|`), como se segue:

```
var pattern:RegExp = /cat|dog/;
// matches cat or dog

pattern = /ca(t|d)og/;
// matches catog or cadog
```

Uso de grupos para capturar correspondências de substrings

Quando terminar de definir um grupo entre parênteses padrão, poderá se referir a ele posteriormente na expressão regular. Isso é conhecido como *referência anterior*, e essas classificações de grupos são conhecidas como *capturas de grupos*. Por exemplo, na expressão regular a seguir, a sequência `\1` corresponde a qualquer substring que correspondeu à captura do grupo entre parênteses:

```
var pattern:RegExp = /(\d+)-by-\1/;
// matches the following: 48-by-48
```

Você pode especificar até 99 dessas referências anteriores em uma expressão regular digitando `\1`, `\2`, ... , `\99`.

De modo semelhante, no método `replace()` da classe String, você pode usar `$1$99`– para inserir correspondência de substring de grupos capturados na string de substituição:

```
var pattern:RegExp = /Hi, (\w+)\./;
var str:String = "Hi, Bob.";
trace(str.replace(pattern, "$1, hello."));
    // output: Bob, hello.
```

Além disso, se você usar a captura de grupo, o método `exec()` da classe RegExp e o método `match()` da classe String retornarão substrings que correspondem à captura de grupos:

```
var pattern:RegExp = /(\w+)@(\w+).(\w+)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
    // bob@example.com,bob,example,com
```

Uso de grupos de não captura e grupos lookahead

Um grupo de não captura é aquele que é usado para agrupamento apenas; ele não é "coletado" e não corresponde a referências anteriores numeradas. Use `(?:` e `)` para definir grupos de não captura, como se segue:

```
var pattern = /(?:com|org|net);
```

Por exemplo, observe a diferença entre colocar `(com|org)` em uma captura versus um grupo de não captura (o método `exec()` lista os grupos de captura depois da conclusão da correspondência):

```
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
```

Um tipo especial de grupo de não captura é o *grupo lookahead*, composto por dois tipos: o *grupo lookahead positivo* e o *grupo lookahead negativo*.

Use `(?=` e `)` para definir um grupo lookahead positivo, que especifica que o subpadrão no grupo deve corresponder à posição. Entretanto, a porção da string que corresponde ao grupo lookahead positivo pode corresponder aos padrões restantes na expressão regular. Por exemplo, como `(?=e)` é um grupo lookahead positivo no código a seguir, o caractere `e` ao qual ele corresponde pode ser correspondido por uma parte subsequente da expressão regular — nesse caso, o grupo de captura, `\w*)`:

```
var pattern:RegExp = /sh(?=e)(\w*)/i;
var str:String = "Shelly sells seashells by the seashore";
trace(pattern.exec(str));
// Shelly,elly
```

Use `(?!` e `)` para definir um grupo lookahead negativo, que especifica que o subpadrão no grupo *não* deve corresponder à posição. Por exemplo:

```
var pattern:RegExp = /sh(?!e)(\w*)/i;
var str:String = "She sells seashells by the seashore";
trace(pattern.exec(str));
// shore,ore
```

Uso de grupos nomeados

Um grupo nomeado é um tipo de grupo em uma expressão regular que tem um identificador nomeado. Use `(?P<name>` e `)` para definir um grupo nomeado. Por exemplo, as expressões regulares a seguir incluem um grupo nomeado com os `dígitos` nomeados do identificador:

```
var pattern = /[a-z]+(?P<digits>\d+)[a-z]+/;
```

Quando você use o método `exec()`, uma correspondência de grupo nomeado é adicionada como uma propriedade da matriz `result`:

```
var myPattern:RegExp = /([a-z]+)(?P<digits>\d+)[a-z]+/;
var str:String = "a123bcd";
var result:Array = myPattern.exec(str);
trace(result.digits); // 123
```

Aqui está outro exemplo, que usa dois grupos nomeados, com os identificadores `name` e `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
```

Nota: *Os grupos nomeados não fazem parte da especificação de linguagem ECMAScript. Eles são um recurso adicionado no ActionScript 3.0.*

Sinalizadores e propriedades

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A tabela a seguir lista os cinco sinalizadores que você pode definir para expressões regulares: Cada sinalizador pode ser acessado como uma propriedade do objeto da expressão regular.

Sinalizador	Propriedade	Descrição
`g`	`global`	Corresponde a mais de uma correspondência.
`i`	`ignoreCase`	Correspondência que não faz distinção entre maiúsculas e minúsculas. Aplica-se aos caracteres `A`—`Z` e `a`—`z`, mas não a caracteres estendidos como `É` e `é`.
`m`	`multiline`	Com esse sinalizador definido, `$` e `^` pode corresponder ao início e ao fim de uma linha, respectivamente.
`s`	`dotall`	Com esse sinalizador definido, `.` (ponto) pode corresponder ao caractere de nova linha (`\n`).
`x`	`extended`	Permite expressões regulares estendidas. Você pode digitar espaços na expressão regular, que serão ignorados como parte do padrão. Isso permite digitar código de expressão regular de modo mais legível.

Observe que essas propriedade são somente leitura. Você pode definir os sinalizadores (`g`, `i`, `m`, `s`, `x`) quando definir uma variável de expressão regular, como se segue:

```
var re:RegExp = /abc/gimsx;
```

Entretanto, não é possível definir diretamente as propriedades nomeadas. Por exemplo, o código a seguir resulta em um erro:

```
var re:RegExp = /abc/;
re.global = true; // This generates an error.
```

Por padrão, a menos que você os especifique na declaração de expressão regular, os sinalizadores não serão definidos e as propriedades de correspondência também são definidas como `false`.

De modo adicional, há duas outras propriedades de uma expressão regular:

- A propriedade `lastIndex` especifica a posição de índice na string para a próxima chamada do método `exec()` or `test()` de uma expressão regular.
- A propriedade `source` especifica a string que define a parte padrão da expressão regular.

O sinalizador g (global)

Quando o sinalizador `g` (`global`) *não* é incluído, uma expressão regular tem não mais do que uma correspondência. Por exemplo, com o sinalizador `g` não incluso na expressão regular, o método `String.match()` retorna somente uma substring de correspondência:

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/;
trace(str.match(pattern)) // output: she
```

Quando o sinalizador `g` é definido, o método `Sting.match()` retorna várias correspondências, como se segue:

```
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
```

O sinalizador i (ignoreCase)

Por padrão, as correspondências de expressões regulares fazem distinção entre maiúsculas e minúsculas. Quando você define o sinalizador `i` (`ignoreCase`), a distinção entre maiúsculas e minúsculas é ignorada. Por exemplo, o `s` minúsculo na expressão regular não corresponde ao `S` maiúsculo, o primeiro caractere da string:

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/)); // output: 13 -- Not the first character
```

Com o sinalizador `i` definido, entretanto, a expressão regular corresponde ao `S` maiúsculo:

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/i)); // output: 0
```

O sinalizador `i` ignora a distinção entre maiúsculas e minúsculas somente para os caracteres `A–Z` e `a–z`, mas não para caracteres estendidos como `É` e `é` .

O sinalizador m (multiline)

Se o sinalizador `m` (`multiline`) não estiver definido, `^` corresponde ao início da string e `$` corresponde ao fim da string. Se o sinalizador `m` estiver definido, esses caracteres corresponderão ao início e ao fim de uma linha, respectivamente. Considere a seguinte string, que inclui um caractere de nova linha:

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/g)); // Match a word at the beginning of the string.
```

Mesmo que o sinalizador `g` (`global`) esteja definido na expressão regular, o método `match()` corresponde a apenas uma substring, pois há apenas uma correspondência para `^` — o início da string. A saída é:

```
Test
```

Aqui está o mesmo código com o sinalizador `m` definido:

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/gm)); // Match a word at the beginning of lines.
```

Neste momento, a saída inclui as palavras no início das linhas:

```
Test,Multiline
```

Observe que apenas o caractere `\n` sinaliza o fim de uma linha. Os caracteres a seguir não:

- Caractere de retorno (`\r`)
- Caractere Unicode separador de linha (`\u2028`)
- Caractere Unicode separador de parágrafo (`\u2029`)

O sinalizador s (dotall)

Se o sinalizador `s` (`dotall` ou "dot all") não estiver definido, um ponto (`.`) em um padrão de expressão regular não corresponde a um caractere de nova linha (`\n`). Portanto para o exemplo a seguir, não há nenhuma correspondência:

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

Entretanto, se o sinalizador `s` estiver definido, o ponto corresponderá ao caractere de nova linha:

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

Nesse caso, a correspondência é a substring inteira dentro das tags `<p>`, incluindo o caractere de nova linha:

```
<p>Test
Multiline</p>
```

O sinalizador x (extended)

As expressões regulares podem ser difíceis de ler, especialmente quando incluem muitos metasímbolos e metassequências. Por exemplo:

```
/<p(>|(\s*[^>]*>)).*?<\/p>/gi
```

Quando você usa o sinalizador `x` (`extended`) em uma expressão regular, qualquer espaço em branco digitado no padrão será ignorado. Por exemplo, a expressão regular a seguir é idêntica ao exemplo anterior:

```
/     <p    (>  |  (\s*  [^>]*  >))    .*?    <\/p>  /gix
```

Se o sinalizador `x` estiver definido e você quiser uma correspondência com o caractere de espaço em branco, preceda o espaço em branco com uma barra invertida. Por exemplo, as duas expressões regulares a seguir são equivalentes:

```
/foo bar/
/foo \ bar/x
```

A propriedade lastIndex

A propriedade `lastIndex` especifica a posição de índice na string no qual a próxima pesquisa será iniciada. Essa propriedade afeta os métodos `exec()` e `test()` chamados em uma expressão regular que tem o sinalizador `g` definido como `true`. Por exemplo, considere o seguinte código:

```
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);
}
```

A propriedade `lastIndex` é definida como 0 por padrão (para iniciar a pesquisa no início da string). Depois de cada correspondência, ela é definida para a posição de índice seguindo a correspondência. Portanto, a saída para o código precedente é a seguinte:

```
0
5
11
18
25
36
44
```

Se o sinalizador `global` estiver definido como `false`, os métodos `exec()` e `test()` não usam nem definem a propriedade `lastIndex`.

Os métodos `match()`, `replace()` e `search()` da classe String iniciam todas as pesquisas no início da string, independentemente da configuração da propriedade `lastIndex` da expressão regular utilizada na chamada do método. (Contudo, o método `match()` define `lastIndex` como 0.)

Você pode definir a propriedade `lastIndex` para ajustar a posição inicial na string para a correspondência da expressão regular.

A propriedade source

A propriedade `source` especifica a string que define a parte padrão de uma expressão regular. Por exemplo:

```
var pattern:RegExp = /foo/gi;
trace(pattern.source); // foo
```

Métodos para usar expressões regulares com strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe RegExp inclui dois métodos: `exec()` e `test()`.

Além dos métodos `exec()` e `test()` da classe RegExp, a classe String inclui os seguintes métodos que permitem corresponder expressões regulares em strings: `match()`, `replace()`, `search()` e `splice()`.

O método test()

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `test()` da classe RegExp verifica simplesmente a string fornecida para ver se ela contém uma correspondência para a expressão regular, como mostra o exemplo a seguir:

```
var pattern:RegExp = /Class-\w/;
var str = "Class-A";
trace(pattern.test(str)); // output: true
```

O método exec()

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `exec()` da classe RegExp verifica a string fornecida quanto a uma correspondência da expressão regular e retorna uma matriz com o seguinte:

- A substring de correspondência
- Correspondência de substring para qualquer grupo entre parênteses na expressão regular

A matriz também inclui uma propriedade `index`, indicando a posição de índice do início da correspondência de substring.

Por exemplo, considere o seguinte código:

```
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
```

Use o método `exec()` várias vezes para corresponder várias substrings quando o sinalizador `g` (`global`) está definido para uma expressão regular:

```
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
```

Métodos String que usam parâmetros RegExp

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os seguintes métodos da classe String usam as expressões regulares como parâmetros: `match()`, `replace()`, `search()` e`split()`. Para obter mais informações sobre esses métodos, consulte "Localização de padrões em sequências de caracteres e substituição de subsequências de caracteres" na página 17.

Exemplo de expressões regulares: um analisador Wiki

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Esse exemplo simples de conversão de texto Wiki ilustra vários usos para expressões regulares:

- Conversão de linhas de texto que correspondem o padrão Wiki de origem às strings de saída HTML apropriadas.
- Uso de uma expressão regular para converter padrões de URL para tags de hiperlinks HTML `<a>`.
- Uso de uma expressão regular para converter strings com dólares norte-americanos (como `"$9,95"`) em strings com euros (como `"8,24 €"`).

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos de aplicativo WikiEditor podem ser encontrados na pasta Samples/WikiEditor. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
WikiEditor.mxml ou WikiEditor.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/regExpExamples/WikiParser.as	Uma classe que inclui métodos que usam expressões regulares para converter padrões de texto de entrada Wiki em saída HTML equivalente.
com/example/programmingas3/regExpExamples/URLParser.as	Uma classe que inclui métodos que usam expressões regulares para converter strings URL para tags de hiperlinks HTML `<a>`.
com/example/programmingas3/regExpExamples/CurrencyConverter.as	Uma classe que inclui métodos que usam expressões regulares para converter strings de dólar americano em strings de euro.

Definição da classe WikiParser

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe WikiParser inclui métodos que convertem texto de entrada Wiki em saída HTML equivalente. Esse não é um aplicativo de conversão Wiki muito robusto, mas ele ilustra alguns bons usos de expressões regulares para correspondência de padrão e conversão de strings.

A função de construtor, junto com o método `setWikiData()`, simplesmente inicializa uma string de amostra do texto de entrada Wiki, como se segue:

```
public function WikiParser()
{
    wikiData = setWikiData();
}
```

Quando o usuário clica no botão Testar no aplicativo de amostra, o aplicativo chama o método `parseWikiString()` do objeto WikiParser. Esse método chama vários outros métodos, que por sua vez montam a string HTML resultante.

```
public function parseWikiString(wikiString:String):String
{
    var result:String = parseBold(wikiString);
    result = parseItalic(result);
    result = linesToParagraphs(result);
    result = parseBullets(result);
    return result;
}
```

Cada um dos métodos chamados — `parseBold()`, `parseItalic()`, `linesToParagraphs()` e `parseBullets()` — usa o método `replace()` da string para substituir os padrões de correspondência, definidos por uma expressão regular, para transformar o texto de entrada Wiki em texto no formato HTML.

Conversão de padrões negrito e itálico

O método `parseBold()` procura padrão de texto negrito Wiki (como `'''foo'''`) e o transforma em seu equivalente em HTML (como `<b>foo</b>`), como se segue:

```
private function parseBold(input:String):String
{
    var pattern:RegExp = /'''(.*?)'''/g;
    return input.replace(pattern, "<b>$1</b>");
}
```

Observe que a parte `(.?*)` de uma pressão regular corresponde a vários caracteres (`*`) entre os dois padrões de definição`'''`. O quantificador `?` torna a correspondência não greedy, para que uma string como `'''aaa''' bbb '''ccc'''`, a primeira string correspondida será `'''aaa'''` e não a string inteira (que começa e termina com o padrão `'''`).

Os parênteses na expressão regular definem um grupo de captura, e o método `replace()` se refere a esse grupo utilizando o código `$1` na string de substituição. O sinalizador `g` (`global`) na expressão regular garante que o método `replace()` substitua todas as correspondências na string (não simplesmente a primeira).

O método `parseItalic()` funciona de forma semelhante ao método `parseBold()`, exceto pelo fato de ele verificar dois apóstrofes (`''`) como o delimitador para texto itálico (não três):

```
private function parseItalic(input:String):String
{
    var pattern:RegExp = /''(.*?)''/g;
    return input.replace(pattern, "<i>$1</i>");
}
```

Conversão de padrões de bullet

Como mostra o exemplo a seguir, o método `parseBullet()` procura o padrão de linha de bullet Wiki (como `* foo`) e o transforma em seu equivalente HTML (como `<li>foo</li>`):

```
private function parseBullets(input:String):String
{
    var pattern:RegExp = /^\*(.*)/gm;
    return input.replace(pattern, "<li>$1</li>");
}
```

O símbolo `^` no início de uma expressão regular corresponde ao início de uma linha. O sinalizador `m` (`multiline`) na expressão regular faz com que essa expressão corresponda o símbolo `^` ao início de uma linha, e não simplesmente ao início da string.

O padrão `\*` corresponde a um caractere asterisco (a barra invertida é usada para sinalizar um asterisco literal em vez de um quantificador `*`).

Os parênteses na expressão regular definem um grupo de captura, e o método `replace()` se refere a esse grupo utilizando o código `$1` na string de substituição. O sinalizador `g` (`global`) na expressão regular garante que o método `replace()` substitua todas as correspondências na string (não simplesmente a primeira).

Conversão de padrões Wiki de parágrafos

O método `linesToParagraphs()` converte cada linha na string Wiki de entrada em uma tag de parágrafo HTML `<p>`. Essas linhas no método retiram linhas vazias da string Wiki de entrada:

```
var pattern:RegExp = /^$/gm;
var result:String = input.replace(pattern, "");
```

Os símbolos `^` e `$` de uma expressão regular correspondem ao início e ao fim de uma linha. O sinalizador `m` (`multiline`) na expressão regular faz com que essa expressão corresponda o símbolo ^ ao início de uma linha, e não simplesmente ao início da string.

O método `replace()` substitui todas as substrings correspondentes (linhas vazias) por uma string vazia (`""`). O sinalizador `g` (`global`) na expressão regular garante que o método `replace()` substitua todas as correspondências na string (não simplesmente a primeira).

Conversão de URLs para tags HTML <a>

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o usuário clica no botão Testar no aplicativo de amostra, se ele marcou a caixa de seleção `urlToATag`, o aplicativo chama o método estático `URLParser.urlToATag()` para converter as strings URL da string Wiki de entrada em tags HTML `<a>`.

```
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>");
```

A função do construtor `RegExp()` é usada para montar uma expressão regular (`urlPattern`) a partir de inúmeras partes constituintes. Essas partes constituintes são cada string que define parte do padrão da expressão regular.

A primeira parte do padrão da expressão regular, definida pela string `protocol`, define um protocolo de URL: `http://` ou `ftp://`. Os parênteses definem um grupo de não captura, indicado pelo símbolo `?`. Isso significa que os parênteses são usados simplesmente para definir um grupo para o padrão de alternação `|`; o grupo não corresponderá a códigos de referência anterior (`$1`, `$2`, `$3`) na string de substituição do método `replace()`.

As outras partes constituintes da expressão regular usam grupos de captura (indicado por parênteses no padrão), que são usados nos códigos de referência anterior (`$1`, `$2`, `$3`) na string de substituição do método `replace()`.

A parte do padrão definido pela string `urlPart` corresponde a *pelo menos* um destes caracteres: `a-z`, `0-9`, _ ou `-`. O quantificador `+` indica que pelo menos um caractere tem correspondência. `\.` indica um caractere de ponto (`.`) exigido. E o restante corresponde a outra string de pelo menos um destes caracteres: `a-z`, `0-9`, _ ou `-`.

A parte do padrão definido pela string `optionalUrlPart` corresponde a *nenhum ou mais* destes caracteres: um ponto (`.`) seguido por qualquer número de caracteres alfanuméricos (incluindo _ e `-`). O quantificador `*` indica que nenhum ou mais caracteres têm correspondência.

A chamada do método `replace()` aplica a expressão regular e monta a string HTML de substituição, utilizando referências anteriores.

O método `urlToATag()` chama o método `emailToATag()`, que usa técnicas semelhantes para substituir padrões de email por string de hiperlinks HTML `<a>`. As expressões regulares utilizadas para corresponder HTTP, FTP e URLs de email nesse arquivo de amostra são muito simples, com o objetivo de exemplificação; há expressões regulares muito mais complicadas para correspondência com esses URLs.

Conversão de strings de dólar americano para strings de euro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o usuário clica no botão Testar do aplicativo de exemplo, se ele marcou a caixa de seleção `dollarToEuro`, o aplicativo chama o método estático `CurrencyConverter.usdToEuro()` para converter as strings com dólares norte-americanos (como `"$9,95"`) em strings com euros (como `"8,24 €"`), da seguinte maneira:

```
var usdPrice:RegExp = /\$([\d,]+.\d+)+/g;
return input.replace(usdPrice, usdStrToEuroStr);
```

A primeira linha define um padrão simples para correspondência de strings de dólar americano. Observe que o caractere `$` é precedido por um caractere escape de barra invertida (`\`).

O método `replace()` usa a expressão regular como o correspondente padrão e chama a função `usdStrToEuroStr()` para determinar a string de substituição (um valor em euros).

Quando um nome de função for utilizado como o parâmetro secundário do método `replace()`, o seguinte será transmitido como parâmetros para a função chamada:

- A parte correspondente da string.
- Qualquer correspondência de grupo em parênteses capturado. O número de argumentos transmitidos dessa maneira irá variar dependendo do número de correspondências de grupo entre parênteses capturado. É possível determinar o número de correspondências de grupo entre parênteses capturado, verificando `arguments.length -3` no código da função.
- A posição de índice na string em que a correspondência começa.
- A string completa.

O método `usdStrToEuroStr()` converte padrões de string de dólar americano para string de euro, como se segue:

```
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;
}
```

Observe que `args[1]` representa o grupo entre parênteses capturado, correspondido pela expressão regular `usdPrice`. Essa é uma parte numérica da string de dólar americano: isto é, a quantidade de dólar sem o sinal `$`. O método aplica uma conversão de taxa de câmbio e retorna a string resultante (com o símbolo € à direita em vez do símbolo $ à esquerda).

Capítulo 6: Trabalho com XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 inclui um grupo de classes com base na especificação ECMAScript para XML (E4X) (ECMA-357 edição 2). Essas classes incluem recursos avançados e fáceis de usar para trabalhar com dados XML. Usando o E4X, você poderá desenvolver códigos com dados XML mais rápido do que com as técnicas de programação anteriores. Além disso, o código produzido será mais fácil de ler.

Mais tópicos da Ajuda

Classe XML

Especificação ECMA-357

Noções básicas sobre XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

XML é uma maneira padrão de representar informações estruturadas com a qual os computadores devem trabalhar com facilidade e que deve ser relativamente fácil para as pessoas gravarem e entenderem. XML é uma abreviação de eXtensible Markup Language (Linguagem de markup extensível). O padrão XML está disponível em www.w3.org/XML/.

O XML oferece um modo prático e padrão de classificar dados, facilitando a leitura, o acesso e a manipulação. O XML usa estruturas de árvore e de tag similares às do HTML. Veja um exemplo simples de dados XML:

```
<song>
    <title>What you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

Os dados XML também podem ser mais complexos, com tags aninhadas em outras tags, bem como em atributos e outros componentes estruturais. Veja um exemplo mais complexo de dados XML:

```
<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>
```

Observe que esse documento XML contém outras estruturas XML completas (como as tags `song` e seus filhos). Ele também demonstra outras estruturas XML como atributos (`tracknumber` e `length` nas tags `song`) e tags que contêm outras tags em vez de dados (como a tag `tracks`).

Como começar a usar o XML

Se você tiver pouca ou nenhuma experiência com XML, veja uma breve descrição dos aspectos mais comuns dos dados XML. Os dados XML são gravados em texto sem formatação, com uma sintaxe específica para organizar as informações em um formato estruturado. Em geral, um único conjunto de dados XML é conhecido como *documento XML*. No formato XML, os dados são organizados em *elementos* (que podem ser itens de dados únicos ou contêineres de outros elementos) usando uma estrutura hierárquica. Cada documento XML tem um único elemento como item de nível superior ou principal; dentro desse elemento raiz, pode existir uma única informação, embora provavelmente haja outros elementos que, por sua vez, contêm outros elementos e assim por diante. Por exemplo, esse documento XML contém as informações sobre um álbum de música:

```
<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>
```

Cada elemento é diferenciado por um conjunto de *tags* – o nome do elemento entre os sinais de menor que e maior que. A tag de abertura, que indica o início do elemento, tem o nome do elemento:

```
<title>
```

A tag de fechamento, que marca o final do elemento, tem uma barra antes do nome do elemento:

```
</title>
```

Se um elemento não tiver nenhum conteúdo, poderá ser gravado como um elemento vazio (às vezes chamado de elemento de fechamento automático). Em XML, esse elemento:

```
<lastplayed/>
```

é idêntico a este elemento:

```
<lastplayed></lastplayed>
```

Além do conteúdo do elemento contido entre as tags de abertura e fechamento, um elemento também pode incluir outros valores, conhecidos como *atributos*, definidos na tag de abertura. Por exemplo, este elemento XML define um único atributo chamado `length`, com o valor `"4:19"` :

```
<song length="4:19"></song>
```

Cada elemento XML tem conteúdo, que pode ser um único valor, um ou mais elementos XML ou nada (para um elemento vazio).

Mais informações sobre XML

Para saber mais sobre como trabalhar com XML, existem diversos outros livros e recursos, incluindo estes sites:

- Tutorial W3Schools XML: http://w3schools.com/xml/
- Tutoriais da XMLpitstop, listas de discussão e muito mais: http://xmlpitstop.com/

Classes do ActionScript para trabalhar com XML

O ActionScript 3.0 inclui várias classes que são usadas para trabalhar com informações estruturadas como XML. As duas classes principais são as seguintes:

- XML: representa um único elemento XML, que pode ser um documento XML com vários filhos ou um elemento com um único valor em um documento.
- XMLList: representa um conjunto de elementos XML. Um objeto XMLList é usado quando existem vários elementos XML que são "irmãos" (no mesmo nível e contidos pelo mesmo pai na hierarquia de documento XML). Por exemplo, uma ocorrência de XMLList seria o modo mais fácil de trabalhar com este conjunto de elementos XML (supostamente contidos em um documento XML):

  ```
  <artist type="composer">Fred Wilson</artist>
  <artist type="conductor">James Schmidt</artist>
  <artist type="soloist">Susan Harriet Thurndon</artist>
  ```

Para usos mais avançados que envolvem namespaces XML, o ActionScript também inclui as classes Namespace e QName. Para obter mais informações, consulte “Uso de namespaces XML” na página 111.

Além das classes internas para trabalhar com XML, o ActionScript 3.0 também inclui vários operadores que fornecem recursos específicos para acessar e manipular dados XML. Essa abordagem de trabalhar com XML usando essas classes e operadores é conhecida como ECMAScript para XML (E4X), conforme definido pela especificação ECMA-357 edição 2.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você vai encontrar ao programar as rotinas de tratamento de XML:

Elemento Um único item em um documento XML, identificado como o conteúdo contido entre uma tag inicial e uma tag final (incluindo as tags). Os elementos XML podem conter dados de texto ou outros elementos, ou podem ser vazios.

Elemento vazio Um elemento XML que não contém nenhum elemento filho. Os elementos vazios geralmente são gravados como tags de fechamento (como `<element/>`).

Documento Uma única estrutura XML. Um documento XML pode conter qualquer número de elementos (ou ser constituído por apenas um único elemento vazio); no entanto, um documento XML deve ter um elemento de nível superior que contém todos os outros elementos do documento.

Nó Outro nome para um elemento XML.

Atributo Um valor nomeado associado a um elemento que está gravado na tag de abertura do elemento no formato `attributename="value"`, em vez de estar gravado como um elemento filho separado aninhado no elemento.

A abordagem E4X em relação ao processamento de XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A especificação ECMAScript para XML define um conjunto de classes e recursos para trabalhar com dados XML. Em conjunto, essas classes e recursos são conhecidos como *E4X*. O ActionScript 3.0 inclui as seguintes classes E4X: XML, XMLList, QName e Namespace.

Os métodos, as propriedades e os operadores das classes E4X foram desenvolvidos com os seguintes objetivos:

- Simplicidade - Sempre que possível, o E4X facilita a gravação e a compreensão do código para trabalhar com dados XML.
- Consistência - Os métodos e princípios por trás do E4X são consistentes internamente e com outras partes do ActionScript.
- Familiaridade – Você manipula os dados XML com operadores conhecidos, como o operador de ponto (.).

Nota: *Há uma classe XML diferente no ActionScript 2.0. No ActionScript 3.0, essa classe foi renomeada como XMLDocument, de modo que não entra em conflito com a classe XML do ActionScript 3.0 que faz parte do E4X. No ActionScript 3.0, as classes herdadas (XMLDocument, XMLNode, XMLParser e XMLTag) são incluídas no pacote flash.xml principalmente para dar suporte a versões anteriores. As novas classes do E4X são classes básicas; não é necessário importar um pacote para utilizá-las. Veja os detalhes das classes antigas do ActionScript 2.0 XML em flash.xml packagena Referência do ActionScript® 3.0 para Adobe® Flash® Platform.*

Veja um exemplo de manipulação dos dados com 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>
```

Normalmente, seu aplicativo carregará dados XML a partir de uma fonte externa, como um serviço da Web ou um feed RSS. No entanto, para fins de clareza, os exemplos de código fornecidos aqui atribuem dados XML como literais.

Como mostra o código a seguir, o E4X inclui alguns operadores intuitivos, como os operadores de ponto (.) e de identificador de atributo (@), para acessar propriedades e atributos no 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
```

Use o método `appendChild()` para atribuir um novo nó filho ao XML, como mostra o snippet a seguir:

```
var newItem:XML =
    <item id="3">
        <menuName>medium cola</menuName>
        <price>1.25</price>
    </item>

myXML.appendChild(newItem);
```

Use os operadores `@` e `.` não só para ler, mas também para atribuir dados do seguinte modo:

```
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";
```

Use um loop `for` para percorrer os nós do XML do seguinte modo:

```
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));
```

Objetos XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um objeto XML pode representar um elemento, atributo, comentários, instrução de processamento ou elemento de texto XML.

Um objeto XML pode ter *conteúdo simples* ou *conteúdo complexo*. Um objeto XML que tem nós filho tem conteúdo complexo. Um objeto XML terá conteúdo simples se contiver um dos seguintes itens: um atributo, um comentário, uma instrução de processamento ou um nó de texto.

Por exemplo, o objeto XML a seguir tem conteúdo complexo, incluindo um comentário e uma instrução de processamento:

```
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>
```

Como mostra o exemplo a seguir, agora você pode usar os métodos `comments()` e `processingInstructions()` para criar novos objetos XML, um comentário e uma instrução de processamento:

```
var x2:XML = x1.comments()[0];
var x3:XML = x1.processingInstructions()[0];
```

propriedades XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe XML tem cinco propriedades estáticas:

- As propriedades `ignoreComments` e `ignoreProcessingInstructions` determinam se comentários ou instruções de processamento devem ser ignorados quando o objeto XML é analisado.
- A propriedade `ignoreWhitespace` determina se os caracteres de espaço em branco devem ser ignorados em tags de elemento e expressões incorporadas que são separadas somente por caracteres de espaço em branco.
- As propriedades `prettyIndenteprettyPrinting` são usadas para formatar o texto que é retornado pelos métodos `toString()` e `toXMLString()` da classe XML.

Veja os detalhes dessas propriedades na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Métodos XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os métodos a seguir permitem trabalhar com a estrutura hierárquica dos objetos XML:

- `appendChild()`
- `child()`
- `childIndex()`
- `children()`
- `descendants()`
- `elements()`
- `insertChildAfter()`
- `insertChildBefore()`

- `parent()`
- `prependChild()`

Os métodos a seguir permitem trabalhar com atributos de objetos XML:

- `attribute()`
- `attributes()`

Os métodos a seguir permitem trabalhar com propriedades de objetos XML:

- `hasOwnProperty()`
- `propertyIsEnumerable()`
- `replace()`
- `setChildren()`

Os métodos a seguir permitem trabalhar com nomes e namespaces qualificados:

- `addNamespace()`
- `inScopeNamespaces()`
- `localName()`
- `name()`
- `namespace()`
- `namespaceDeclarations()`
- `removeNamespace()`
- `setLocalName()`
- `setName()`
- `setNamespace()`

Os métodos a seguir permitem trabalhar e determinar tipos específicos de conteúdo XML:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `nodeKind()`
- `processingInstructions()`
- `texto()`

Os métodos a seguir servem para conversão em strings e formatação de objetos XML:

- `defaultSettings()`
- `setSettings()`
- `settings()`
- `normalize()`
- `toString()`
- `toXMLString()`

Existem alguns métodos adicionais:

- `contains()`
- `copiar()`
- `valueOf()`
- `length()`

Veja a documentação desses métodos na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Objetos XMLList

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma ocorrência de XMLList representa uma coleção arbitrária de objetos XML. Ela contém documentos XML completos, fragmentos de XML ou os resultados de uma consulta XML.

Os métodos a seguir permitem trabalhar com a estrutura hierárquica dos objetos XMLList:

- `child()`
- `children()`
- `descendants()`
- `elements()`
- `parent()`

Os métodos a seguir permitem trabalhar com atributos de objetos XMLList:

- `attribute()`
- `attributes()`

Os métodos a seguir permitem trabalhar com propriedades de XMLList:

- `hasOwnProperty()`
- `propertyIsEnumerable()`

Os métodos a seguir permitem trabalhar e determinar tipos específicos de conteúdo XML:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `processingInstructions()`
- `texto()`

Os métodos a seguir servem para conversão em strings e formatação do objeto XMLList:

- `normalize()`
- `toString()`
- `toXMLString()`

Existem alguns métodos adicionais:

- `contains()`

- `copiar()`
- `length()`
- `valueOf()`

Veja a documentação desses métodos na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Para um objeto XMLList que contém exatamente um elemento XML, você pode usar todos os métodos e propriedades da classe XML porque um XMLList com um elemento XML é tratado do mesmo modo que um objeto XML. Por exemplo, no código a seguir, como `doc.div` é um objeto XMLList que contém um elemento, você pode usar o método `appendChild()` da classe XML:

```
var doc:XML =
        <body>
            <div>
                <p>Hello</p>
            </div>
        </body>;
doc.div.appendChild(<p>World</p>);
```

Para obter uma lista de propriedades e métodos XML, consulte "Objetos XML" na página 100.

Inicialização de variáveis XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode atribuir um literal XML a um objeto XML do seguinte modo:

```
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>
```

Como mostra o snippet a seguir, também é possível usar o construtor `new` para criar uma ocorrência de um objeto XML a partir de uma string que contém dados XML:

```
var str:String = "<order><item id='1'><menuName>burger</menuName>"
                           + "<price>3.95</price></item></order>";
var myXML:XML = new XML(str);
```

Se os dados XML da string não estiverem bem formados (por exemplo, se estiver faltando uma tag de fechamento), ocorrerá um erro de tempo de execução.

Você também pode transmitir os dados por referência (de outras variáveis) em um objeto XML, como mostra o exemplo a seguir:

```
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>
```

Para carregar os dados XML a partir de uma URL, use a classe URLLoader, como mostra o exemplo a seguir:

```
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!");
    }
}
```

Para ler dados XML a partir de uma conexão de soquete, use a classe XMLSocket. Para obter mais informações, consulte a classe XMLSocket na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Montagem e transformação de objetos XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use o método `prependChild()` ou o método `appendChild()` para adicionar uma propriedade ao início ou ao final de uma lista de propriedades do objeto XML, como mostra o exemplo a seguir:

```
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>
```

Use o método `insertChildBefore()` ou o método `insertChildAfter()` para adicionar uma propriedade antes ou depois de uma propriedade especificada, do seguinte modo:

```
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>)
```

Como mostra o exemplo a seguir, você também pode usar os operadores de chaves ({ e }) para possar dados por referência (de outras variáveis) ao criar objetos XML:

```
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)
}
```

É possível atribuir propriedades e atributos a um objeto XML usando o operador `=`, conforme mostrado a seguir:

```
var x:XML =
    <employee>
        <lastname>Smith</lastname>
    </employee>
x.firstname = "Jean";
x.@id = "239";
```

Isso define o objeto XML `x` como:

```
<employee id="239">
    <lastname>Smith</lastname>
    <firstname>Jean</firstname>
</employee>
```

Você pode usar os operadores + e += para concatenar objetos XMLList:

```
var x1:XML = <a>test1</a>
var x2:XML = <b>test2</b>
var xList:XMLList = x1 + x2;
xList += <c>test3</c>
```

Isso define o objeto XMLList `xList` como:

```
<a>test1</a>
<b>test2</b>
<c>test3</c>
```

Como percorrer estruturas XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um dos recursos avançados do XML é fornecer dados aninhados complexos por meio de uma string linear de caracteres de texto. Ao carregar dados em um objeto XML, o ActionScript analisa os dados e carrega sua estrutura hierárquica na memória (ou envia um erro de tempo de execução se os dados XML não estiverem bem formados).

Os operadores e métodos dos objetos XML e XMLList facilitam o percurso pela estrutura de dados XML.

Use o operador de dot (.) e o operador de acessador do descendente (..) para acessar as propriedades filho de um objeto XML. Considere o objeto XML a seguir:

```
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>
```

O objeto `myXML.book` é um objeto XMLList que contém as propriedades filhas do objeto `myXML` chamado `book`. Esses dois objetos XML correspondem às duas propriedades `book` do objeto `myXML`.

O objeto `myXML..lastName` é um objeto XMLList que contém todas as propriedades do descendente com o nome `lastName`. Esses dois objetos XML correspondem às duas propriedades `lastName` do objeto `myXML`.

O objeto `myXML.book.editor.lastName` é um objeto XMLList que contém todos os filhos com o nome `lastName` dos filhos com o nome `editor` dos filhos com o nome `book` do objeto `myXML`: nesse caso, um objeto XMLList que contém apenas um objeto XML (a propriedade `lastName` com o valor "`Case`").

Acesso a nós pai e filho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `parent()` retorna o pai de um objeto XML.

Você pode usar os valores ordinais de índice de uma lista de filhos para acessar objetos filho específicos. Por exemplo, considere um objeto XML `myXML` que tem duas propriedades filho chamadas `book`. Cada propriedade filho chamada `book` tem um número de índice associado:

```
myXML.book[0]
myXML.book[1]
```

Para acessar um neto específico, você pode especificar números de índice para os nomes do filho e do neto:

```
myXML.book[0].title[0]
```

No entanto, se houver apenas um filho de `x.book[0]` com o nome `title`, você pode omitir a referência de índice do seguinte modo:

```
myXML.book[0].title
```

Do mesmo modo, se houver apenas um filho de book do objeto `x`, e se esse objeto filho tiver apenas um objeto title, você pode omitir as duas referências de índice assim:

```
myXML.book.title
```

É possível usar o método `child()` para navegar até os filhos com nomes baseados em uma variável ou expressão, como mostra o exemplo a seguir:

```
var myXML:XML =
        <order>
            <book>
                <title>Dictionary</title>
            </book>
        </order>;

var childName:String = "book";

trace(myXML.child(childName).title) // output: Dictionary
```

Acesso a atributos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use o símbolo `@` (operador de identificador de atributo) para acessar atributos em um objeto XML ou XMLList, como mostra o código a seguir:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@id); // 6401
```

Você pode usar o símbolo de caractere curinga `*` com o símbolo `@` para acessar todos os atributos de um objeto XML ou XMLList, como no código a seguir:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@*.toXMLString());
// 6401
// 233
```

Você pode usar o método `attribute()` ou `attributes()` para acessar um atributo específico ou todos os atributos de um objeto XML ou XMLList, como no código a seguir:

```
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
```

Também é possível usar a sintaxe a seguir para acessar atributos, como mostra o seguinte exemplo:

```
employee.attribute("id")
employee["@id"]
employee.@["id"]
```

Cada um é equivalente a `employee.@id`. No entanto, a sintaxe `employee.@id` é recomendada.

Filtragem por valor de elemento ou atributo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar os operadores de parênteses – `(` e `)` – para filtrar elementos com um nome de elemento ou valor de atributo específico. Considere o objeto XML a seguir:

```
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>
```

As seguintes expressões são válidas:

- `x.employee.(lastName == "McGee")` – É o segundo nó `employee`.
- `x.employee.(lastName == "McGee").firstName` – É a propriedade `firstName` do segundo nó `employee`.
- `x.employee.(lastName == "McGee").@id` – É o valor do atributo `id` do segundo nó `employee`.
- `x.employee.(@id == 347)` – O primeiro nó `employee`.
- `x.employee.(@id == 347).lastName` – É a propriedade `lastName` do primeiro nó `employee`.
- `x.employee.(@id > 300)` – É um XMLList com as duas propriedades `employee`.
- `x.employee.(position.toString().search("analyst") > -1)` – É um objeto XMLList com as duas propriedades `position`.

Se você tentar filtrar os atributos ou elementos que não existem, é lançada uma exceção. Por exemplo, a linha final do código a seguir gera um erro, porque não existe nenhum atributo `id` no segundo elemento `p`:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(@id == '123'));
```

Do mesmo modo, a linha final do código a seguir gera um erro, porque não existe nenhuma propriedade `b` do segundo elemento `p`:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(b == 'Bob'));
```

Para evitar esses erros, você pode identificar as propriedades que têm atributos ou elementos correspondentes usando os métodos `attribute()` e `elements()`, como no código a seguir:

```
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'));
```

Também é possível usar o método `hasOwnProperty()`, como no seguinte código:

```
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'));
```

Uso de for..in e for each..em instruções

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 inclui a instrução `for..in` e a instrução `for each..in` para percorrer objetos XMLList. Por exemplo, considere o seguinte objeto XML, `myXML`, e o objeto XMLList, `myXML.item`. O objeto XMLList, `myXML.item`, consiste em dois nós `item` do objeto XML.

```
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>;
```

A instrução `for..in` permite percorrer um conjunto de nomes de propriedades em um XMLList:

```
var total:Number = 0;
for (var pname:String in myXML.item)
{
    total += myXML.item.@quantity[pname] * myXML.item.price[pname];
}
```

A instrução `for each..in` permite percorrer as propriedades em um objeto XMLList:

```
var total2:Number = 0;
for each (var prop:XML in myXML.item)
{
    total2 += prop.@quantity * prop.price;
}
```

Uso de namespaces XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os namespaces em um objeto (ou documento) XML identificam o tipo de dados contido no objeto. Por exemplo, ao enviar e fornecer dados XML para um serviço da Web que usa o protocolo SOAP, você declara o namespace na tag de abertura do XML:

```
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>;
```

O namespace tem um prefixo, `soap`, e uma URI que define o namespace, `http://schemas.xmlsoap.org/soap/envelope/`.

O ActionScript 3.0 inclui a classe Namespace para trabalhar com namespaces XML. Para o objeto XML do exemplo anterior, você pode usar a classe Namespace do seguinte modo:

```
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";
```

A classe XML inclui os seguintes métodos para trabalhar com namespaces: `addNamespace()`, `inScopeNamespaces()`, `localName()`, `name()`, `namespace()`, `namespaceDeclarations()`, `removeNamespace()`, `setLocalName()`, `setName()` e `setNamespace()`.

A diretiva de `namespace XML padrão` permite atribuir um namespace padrão a objetos XML. Por exemplo, a seguir, `x1` e `x2` têm o mesmo namespace padrão:

```
var ns1:Namespace = new Namespace("http://www.example.com/namespaces/");
default xml namespace = ns1;
var x1:XML = <test1 />;
var x2:XML = <test2 />;
```

Conversão de tipo XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode converter objetos XML e XMLList em valores de string. Do mesmo modo, é possível converter strings em objetos XML e XMLList. Além disso, tenha em mente que todos os valores de atributo, nomes e valores de texto XML são strings. As seções a seguir discutem todas essas formas de conversão de tipo XML.

Conversão de objetos XML e XMLList em strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As classes XML e XMLList incluem um método `toString()` e um método `toXMLString()`. O método `toXMLString()` retorna uma string que inclui todas as tags, atributos, instruções de namespace e conteúdo do objeto XML. Para objetos XML com conteúdo complexo (elementos filho), o método `toString()` é exatamente igual ao método `toXMLString()`. Para objetos XML com conteúdo simples (aqueles que contêm apenas um elemento de texto), o método `toString()` retorna somente o conteúdo de texto do elemento, como mostra o exemplo a seguir:

```
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
```

Se o método `trace()` é usado sem especificar `toString()` ou `toXMLString()`, os dados são convertidos usando o método `toString()` por padrão, como mostra este código:

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName);
    // burger
```

Ao usar o método `trace()` para depurar o código, você normalmente usará o método `toXMLString()` para que `trace()` gere dados mais completos.

Conversão de strings em objetos XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar o construtor `new XML()` para criar um objeto XML a partir de uma string, do seguinte modo:

```
var x:XML = new XML("<a>test</a>");
```

Se você tentar converter uma string em XML a partir de uma string que representa um XML inválido ou mal formado, ocorrerá um erro de tempo de execução do seguinte modo:

```
var x:XML = new XML("<a>test"); // throws an error
```

Conversão de valores de atributo, nomes e valores de texto a partir de strings

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Todos os valores de atributo, nomes e valores de texto XML são tipos de dados String e talvez seja necessário convertê-los em outros tipos de dados. Por exemplo, o código a seguir usa a função `Number()` para converter valores de texto em números:

```
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;
```

Se esse código não tivesse usado a função `Number()`, o código interpretaria o operador + como o operador de concatenação de string e o método `trace()` na última linha seria o seguinte:

```
01.003.95
```

Leitura de documentos XML externos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar a classe URLLoader para carregar dados XML a partir de uma URL. Para usar o código a seguir em seus aplicativos, substitua o valor `XML_URL` do exemplo por uma URL válida:

```
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.");
}
```

Você também pode usar a classe XMLSocket para configurar uma conexão de soquete XML assíncrona com um servidor. Para obter mais informações, consulte Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Exemplo XML no ActionScript: carregamento de dados RSS da Web

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo de exemplo RSSViewer mostra diversos recursos para trabalhar com XML no ActionScript, incluindo os seguintes:

- Uso dos métodos XML para percorrer dados XML em forma de um feed RSS.
- Uso dos métodos XML para montar dados XML em forma de HTML a ser usado em um campo de texto.

O formato RSS é muito utilizado para distribuir notícias via XML. Um arquivo de dados RSS simples pode ser parecido com o seguinte:

```
<?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>
```

O aplicativo SimpleRSS lê os dados RSS na Internet, analisa os dados em busca de cabeçalhos (títulos), links e descrições e retorna esses dados. A classe SimpleRSSUI fornece a interface de usuário e chama a classe SimpleRSS, que faz todo o processamento XML.

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo RSSViewer estão localizados na pasta Amostras/RSSViewer. O aplicativo consiste nos seguintes arquivos:

Arquivo	**Descrição**
RSSViewer.mxml ou RSSViewer.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/rssViewer/RSSParser.as	Uma classe que contém métodos que usam o E4X para percorrer dados RSS (XML) e gerar uma representação em HTML correspondente.
RSSData/ak.rss	Um arquivo RSS de exemplo. O aplicativo é configurado para ler dados RSS na Web, em um feed RSS do Flex hospedado pela Adobe. No entanto, você pode alterar o arquivo com facilidade para ler dados RSS neste documento, que usa um esquema ligeiramente diferente do feed RSS do Flex.

Leitura e análise de dados XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe RSSParser inclui um método `xmlLoaded()` que converte os dados RSS de entrada, armazenados na variável `rssXML`, em uma string que contém a saída em formato HTML, `rssOutput`.

Logo do início do método, o código define o namespace XML padrão se os dados RSS de origem incluírem um namespace padrão:

```
if (rssXML.namespace("") != undefined)
{
    default xml namespace = rssXML.namespace("");
}
```

As próximas linhas percorrem o conteúdo dos dados XML de origem, examinando cada propriedade de descendente chamada `item`:

```
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);
}
```

As três primeiras linhas simplesmente definem variáveis de string para representar as propriedades de título, descrição e link da propriedade `item` dos dados XML. Em seguida, a próxima linha chama o método `buildItemHTML()` para obter os dados HTML em forma de um objeto XMLList, usando as três novas variáveis de string como parâmetros.

Montagem de dados XMLList

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os dados HTML (um objeto XMLList) têm uma das seguintes formas:

```
<b>itemTitle</b>
<p>
    itemDescription
    <br />
    <a href="link">
        <font color="#008000">More...</font>
    </a>
</p>
```

As primeiras linhas do método apagam o namespace XML padrão:

```
default xml namespace = new Namespace();
```

A diretiva de `namespace XML padrão` tem o escopo do nível de bloqueio da função. Isso significa que os escopo dessa instrução é o método `buildItemHTML()`.

As próximas linhas montam o XMLList, com base nos argumentos de string transmitidos para a função:

```
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;
```

Esse objeto XMLList representa dados de string adequados para um campo de texto HTML do ActionScript.

O método `xmlLoaded()` usa o valor de retorno do método `buildItemHTML()` e o converte em uma string:

```
XML.prettyPrinting = false;
rssOutput = outXML.toXMLString();
```

Extração do título do feed RSS e envio de um evento personalizado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `xmlLoaded()` define uma variável de string `rssTitle`, com base nas informações dos dados XML RSS de origem:

```
rssTitle = rssXML.channel.title.toString();
```

Finalmente, o método `xmlLoaded()` gera um evento, que informa ao aplicativo que os dados estão analisados e disponíveis:

```
            dataWritten = new Event("dataWritten", true);
```

Capítulo 7: Uso da funcionalidade JSON nativa

O ActionScript 3.0 fornece uma API nativa para objetos de codificação e decodificação dos objetos ActionScript que usam o formato JavaScript Object Notation (JSON). A classe JSON e as funções de suporte ao membro seguem a especificação da 5ª edição da ECMA-262 com algumas variações.

O membro da comunidade Todd Anderson fornece uma comparação entre a API JSON nativa e a classe as3corelib JSON de terceiros. Consulte Trabalhando com o JSON Nativo no Flash Player 11.

Mais tópicos da Ajuda

JSON

Visão geral da API JSON

A API JSON do ActionScript consiste na classe JSON e nas funções de membro `toJSON()` em algumas classes nativas. Para os aplicativos que exigem uma codificação JSON personalizada para qualquer classe, a estrutura do ActionScrip fornece maneiras de sobrescrever a codificação padrão.

A classe JSON manipula internamente a importação e a exportação de qualquer classe do ActionScript que não forneça um membro `toJSON()`. Para esses casos, o JSON cruza as propriedades públicas de cada objeto que encontra. Caso um objeto contenha outros objetos, a JSON usará a função recursiva nos objetos aninhados e realizará o mesmo cruzamento. Caso qualquer objeto forneça um método `toJSON()`, a JSON usará aquele método customizado em vez de seu algoritmo interno.

A interface JSON consiste em um método de codificação, `stringify()`, e em um método de decodificação, `parse()`. Cada um desses métodos fornece um parâmetro que permite a inserção de sua própria lógica no fluxo de trabalho de codificação e decodificação do JSON. Para `stringify()`, esse parâmetro é denominado `replacer`; para `parse()`, o parâmetro é `reviver`. Esses parâmetros assumem a definição da função com dois argumentos que usam a seguinte assinatura:

```
function(k, v):*
```

métodos toJSON()

A assinatura para métodos `toJSON()` é

```
public function toJSON(k:String):*
```

`JSON.stringify()` chamará `toJSON()` se ele existir, em cada propriedade pública que encontrar durante o cruzamento de um objeto. Uma propriedade consiste em um par de chave e valor. Quando `stringify()` executar `toJSON()`, enviará a chave `k`da propriedade que está examinando no momento. Uma implementação típica`toJSON()` avalia cada nome de propriedade e retorna a decodificação desejada de seu valor.

O método `toJSON()` pode retornar um valor de qualquer tipo (denominado *)— não somente uma Sequência de caracteres. Esse tipo de retorno variável permite que `toJSON()` retorne um objeto, se for o caso. Por exemplo, caso uma propriedade de sua classe customizada contenha um objeto de outra biblioteca de terceiros, você poderá retornar aquele objeto quando `toJSON()` encontrar sua propriedade. A classe JSON então usará a função recursiva no objeto de terceiros. O fluxo do processo de codificação apresenta o seguinte comportamento:

- Caso `toJSON()` retorne um objeto que não seja avaliado como uma string, `stringify()` usará a função recursiva naquele objeto.
- Se `toJSON()` retornar uma string, `stringify()` envolve esse valor em outra string, retorna a string envolvida e, em seguida, move-se para o próximo valor.

Em muitos casos, retornar um objeto é preferível a retornar uma string JSON criada pelo seu aplicativo. Retornar um objeto aproveita o algoritmo de codificação JSON e também permite que JSON usar a função recursiva em objetos aninhados.

O método `toJSON()` não está definido na classe Object nem na maioria das outras classes nativas. Sua ausência indica a JSON que execute sua travessia padrão nas propriedades públicas do objeto. Se desejar, você também pode usar `toJSON()` para exibir as propriedades privadas do objeto.

Algumas classes nativas representam um desafio que as bibliotecas do ActionScript não conseguem resolver de maneira eficaz para todos os casos de uso. Para essas classes, o ActionScript fornece uma implementação comum que os clientes podem reimplementar para se adaptarem a suas necessidades. A classe que fornece membros `toJSON()` triviais incluem:

- ByteArray
- Date
- Dictionary
- XML

Você pode classificar em uma subclasse a classe ByteArray para sobrescrever o método `toJSON()` ou pode redefinir seu protótipo. As classes Date e XML, que são declaradas como finais, exigem a utilização de um protótipo de classe para redefinir `toJSON()`. A classe Dictionary é declarada dinamicamente, o que permite a liberdade adicional na substituição de `toJSON()`.

Definição do comportamento personalizado de JSON

Para implementar sua própria codificação e decodificação JSON para classes nativas, você pode optar por várias alternativas:

- Definição ou substituição de `toJSON()` em sua subclasse personalizada de uma classe nativa que não seja final
- Definição ou redefinição de `toJSON()` no protótipo da classe
- Definição de uma propriedade de `toJSON` em uma classe dinâmica
- Uso dos parâmetros `JSON.stringify() replacer` e `JSON.parser() reviver`

Mais tópicos da Ajuda

ECMA-262, 5ª Edição

Definição de toJSON() no protótipo de uma classe integrada

A implementação nativa de JSON no ActionScript reflete o mecanismo de JSON ECMAScript , definido em ECMA-262, 5ª edição. Uma vez que ECMAScript não suporta classes, o ActionScript define o comportamento de JSON em termos de despacho baseado em protótipo. Protótipos são precursores das classes do ActionScript 3.0 que permitem a herança simulada, bem como adições e redefinições de membros.

O ActionScript permite a definição ou redefinição de `toJSON()` no protótipo de qualquer classe. Esse privilégio é aplicado até para classes que são marcadas como finais. Ao definir `toJSON()` em um protótipo de classe, sua definição se torna atual para todas as instâncias daquela classe dentro do escopo de sua aplicação. Por exemplo, essa é uma maneira de definir um método `toJSON()` no protótipo MovieClip:

```
MovieClip.prototype.toJSON = function(k):* {
    trace("prototype.toJSON() called.");
    return "toJSON";
}
```

Quando, em seguida, o aplicativo chamar`stringify()` em qualquer instância do MovieClip, o `stringify()` retornará a saída de seu método `toJSON()`:

```
var mc:MovieClip = new MovieClip();
var js:String = JSON.stringify(mc); //"prototype toJSON() called."
trace("js: " + js); //"js: toJSON"
```

Também é possível substituir `toJSON()` nas classes nativas que definem o método. Por exemplo, o seguinte código substitui `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"
```

Definição ou substituição de toJSON() no nível de classe

Não é necessário que os aplicativos usem protótipos para redefinir `toJSON()`. Você também pode definir `toJSON()` como membro de uma subclasse se a classe pai não estiver marcada como final. Por exemplo, você pode estender a classe ByteArray e definir uma função `toJSON()` pública:

```
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"
```

Se a classe for dinâmica, você poderá adicionar uma propriedade `toJSON` a um objeto daquela classe e designar uma função para ela como segue:

```
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."}
```

É possível substituir, definir ou redefinir `toJSON()` em qualquer classe do ActionScript. No entanto, a maioria das classes incorporadas do ActionScript não definem `toJSON()`. A classe Object não define `toJSON` em seu protótipo padrão nem o declara como um membro da classe. Somente algumas poucas classes nativas definem o método como uma função de protótipo. Dessa forma, na maioria das classes não é possível substituir `toJSON()` da maneira tradicional.

As classes nativas que não definem `toJSON()` são serializadas em JSON pela implementação interna de JSON. Se possível, evite substituir essa funcionalidade incorporada. Se você definir um membro de `toJSON()`, a classe JSON usará sua lógica em vez de sua própria funcionalidade.

Uso do parâmetro replacer JSON.stringify()

A substituição de `toJSON()` no protótipo é útil nas alterações de comportamento de exportação de JSON de uma classe num aplicativo. Em alguns casos, sua lógica de exportação poderá ser aplicada somente para casos especiais sob condições temporárias. Para acomodar essas alterações de escopo tão pequeno, use o parâmetro `replacer` do método `JSON.stringify()`.

O método `stringify()` aplica a função transmitida ao parâmetro `replacer` para o objeto a ser decodificado. A assinatura para essa função é similar à de `toJSON()`:

```
function (k,v):*
```

Diferente de `toJSON()`, a função `replacer` exige o valor `v` e a chave, `k`. Essa diferença é necessária porque `stringify()` é definido no objeto JSON estático em vez de no objeto a ser decodificado. Quando `JSON.stringify()` chamar`replacer(k,v)`, ele estará cruzando o objeto de entrada original. O parâmetro implícito `this` enviado à função `replacer` refere-se ao objeto que possui a chave e o valor. Uma vez que `JSON.stringify()` não altera o objeto de entrada original, aquele objeto permanece inalterado no container a ser cruzado. Dessa forma, é possível usar o código`this[k]` para consultar a chave no objeto original. O parâmetro `v` mantém o valor que `toJSON()` converteu.

Como `toJSON()`, a função `replacer` pode retornar qualquer tipo de valor. Se `replacer` retornar uma sequência de caracteres, o mecanismo JSON escapará os conteúdos em citações e, em seguida, agrupará também os conteúdos de escape em citações. Esse agrupamento garante que `stringify()` receba um objeto de sequência de caracteres JSON válido que permaneça como uma sequência de caracteres em uma chamada subsequente para `JSON.parse()`.

O seguinte código usa o parâmetro `replacer` e o parâmetro implícito `this` para retornar os valores de `time` e `hours` de um objeto Date:

```
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;
});
```

Utilização do parâmetro reviver JSON.parse()

O parâmetro `reviver` do método `JSON.parse()` realiza o oposto da função `replacer` : ele converte uma sequência de caracteres JSON em um objeto do ActionScript utilizável. O argumento `reviver` é uma função que usa dois parâmetros e retorna qualquer tipo:

```
function (k,v):*
```

Nessa função, `k` é uma chave, e `v` é o valor de `k`. Como `stringify()`, o`parse()` cruza os pares de valor-chave de JSON e aplica a função `reviver` - caso ela exista - a cada par. Um problema em potencial é o fato de a classe JSON não fornecer a saída de um nome de classe do ActionScript de um objeto. Assim, pode ser difícil saber que tipo de objeto deve ser revivido. Esse problema pode ser especialmente grave quando os objetos são aninhados. Ao designar as funções `toJSON()`, `replacer` e `reviver`, você pode planejar maneiras de identificar os objetos do ActionScript que são exportados, enquanto mantém os objetos originais intactos.

Exemplo de análise

O seguinte exemplo mostra uma estratégia para reviver objetos analisados de sequências de caracteres do JSON. Este exemplo define duas classes: JSONGenericDictExample e JSONDictionaryExtnExample. A classe JSONGenericDictExample é uma classe de dicionário personalizado. Cada registro contém o nome e a data de aniversário de uma pessoa, assim como uma ID exclusiva. Cada vez que o construtor JSONGenericDictExample é chamado, ele adiciona o objeto recém-criado a uma matriz estática interna com um número inteiro com incremento estático como sua ID. A classe JSONGenericDictExample também define um método `revive()` que extrai somente a parte correspondente ao número inteiro do membro de `id` mais longo. O método `revive()` usa esse número inteiro para buscar e retornar o objeto restaurável correto.

A classe JSONDictionaryExtnExample estende a classe Dictionary do ActionScript. Seus registros não têm uma estrutura definida e podem conter quaisquer dados. Os dados são atribuídos depois que o objeto JSONDictionaryExtnExample é construído, e não como propriedades definidas pela classe. Registros JSONDictionaryExtnExample usam objetos JSONGenericDictExample como chaves. Quando um objeto JSONDictionaryExtnExample é revivido, a função `JSONGenericDictExample.revive()` usa a ID associada a JSONDictionaryExtnExample para recuperar o objeto de chave correto.

O mais importante é que o método `JSONDictionaryExtnExample.toJSON()` retorna uma string de marcador além do objeto JSONDictionaryExtnExample. Essa string identifica a saída de JSON como pertencente à classe JSONDictionaryExtnExample. Esse marcador não deixa dúvidas quanto ao tipo de objeto que está sendo processado durante `JSON.parse()`.

```
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;
        }
    }
}
```

Quando o script de tempo de execução a seguir chama `JSON.parse()` em um objeto JSONDictionaryExtnExample, a função `reviver` chama `JSONGenericDictExample.revive()` em cada objeto em JSONDictionaryExtnExample. Essa chamada extrai a ID que representa a chave do objeto. A função `JSONGenericDictExample.revive()` usa essa ID para recuperar e retornar o objeto JSONDictionaryExtnExample armazenado em uma matriz estática privada.

```
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]>]
```

Capítulo 8: Manipulação de eventos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O sistema de tratamento de eventos permite que os programadores respondam às entradas de usuários e eventos do sistema de forma conveniente. O modelo de evento do ActionScript 3.0 não é apenas conveniente, mas também compatível com os padrões, e bem integrado com a lista de exibição. Com base na especificação de eventos DOM (Document Object Model) de nível 3 e em uma arquitetura de manipulação de eventos padrão do setor, o novo modelo de evento fornece uma ferramenta poderosa e intuitiva para os programadores do ActionScript.

O sistema de tratamento de eventos do ActionScript 3.0 interage de perto com a lista de exibição. Para ter uma compreensão básica da lista de exibição, leia “Programação de exibição” na página 153.

Mais tópicos da Ajuda

Pacote flash.events

Especificação de eventos DOM (Document Object Model) de nível 3

Noções básicas sobre a manipulação de eventos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Pense nos eventos como ocorrências de qualquer tipo no arquivo SWF que interessam a você como programador. Por exemplo, a maioria dos arquivos SWF oferece suporte a algum tipo de interação do usuário - seja algo simples, como responder ao clique do mouse, ou algo mais complexo, como aceitar e processar dados inseridos em um formulário. Toda interação do usuário com seu arquivo SWF é considerada um evento. Os eventos também podem ocorrer sem nenhuma interação direta do usuário, como quando os dados terminam de ser carregados a partir de um servidor ou quando uma câmera acoplada é ativada.

No ActionScript 3.0, cada evento é representado por um objeto, que é uma ocorrência da classe Event ou uma de suas subclasses. O objeto de evento não apenas armazena informações sobre um evento específico, mas também contém métodos que facilitam o próprio tratamento. Por exemplo, quando detecta um clique do mouse, o Flash Player ou o AIR cria um objeto de evento (uma ocorrência da classe MouseEvent) para representar esse evento específico de clique do mouse.

Depois de criar um objeto de evento, o Flash Player ou o AIR o *envia*, ou seja, o objeto de evento é transmitido para o objeto que é destino do evento. O objeto que é o destino de um objeto de evento enviado é chamado de *destino do evento*. Por exemplo, quando uma câmera acoplada é ativada, o Flash Player envia um objeto de evento diretamente ao destino que, nesse caso, é o objeto que representa a câmera. No entanto, se o destino do evento estiver na lista de exibição, o objeto será transmitido pela hierarquia da lista de exibição até atingir o destino do evento. Em alguns casos, o objeto de evento forma "bolhas" na hierarquia da lista de exibição ao longo da mesma rota. Essa profundidade da hierarquia da lista de exibição é chamada de *fluxo de evento*.

Você pode “ouvir” objetos de evento no seu código usando ouvintes de evento. *Ouvintes de evento* são as funções ou os métodos gravados para responder a eventos específicos. Para garantir que seu programa responda a eventos, adicione ouvintes ao destino do evento ou a qualquer objeto da lista de exibição que faça parte do fluxo de um objeto de evento.

Sempre que é gravado, o código do ouvinte de evento segue essa estrutura básica (os elementos em negrito são alocadores de espaço preenchidos de acordo com suas necessidades):

```
function eventResponse(eventObject:EventType):void
{
    // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

Esse código faz duas coisas. Primeiro, ele define uma função, que é a maneira de especificar as ações que serão executadas em resposta ao evento. Em seguida, o método `addEventListener()` do objeto de origem é chamado, basicamente "inscrevendo" a função do evento especificado para que, quando o evento acontecer, as ações da função sejam executadas. Quando o evento realmente acontece, o destino do evento verifica a lista de todos os métodos e funções registrados como ouvintes de evento. Em seguida, ele chama um por um, passando o objeto de evento como parâmetro.

Para criar seu próprio ouvinte de evento, é necessário alterar quatro coisas nesse código. Primeiro, você deve dar à função o nome que deseja usar (essa alteração deve ser feita em dois lugares, onde aparece **eventResponse** no código). Segundo, você deve especificar o nome de classe adequado do objeto que é enviado pelo evento que deseja ouvir (**EventType** no código) e também deve especificar a constante correta para o evento em questão (**EVENT_NAME** na listagem). Terceiro, você deve chamar o método `addEventListener()` no objeto que enviará o evento (**eventTarget** neste código). Se desejar, altere o nome da variável usada como parâmetro da função (**eventObject** neste código).

Conceitos e termos importantes

A lista a seguir de referência contém termos importantes que você vai encontrar ao escrever rotinas de tratamento de eventos:

Bolhas As bolhas ocorrem para alguns eventos, de forma que um objeto de exibição pai possa responder aos eventos despachados por seus filhos.

Fase de bolhas A parte do fluxo de evento no qual um evento se propaga até os objetos de exibição pai. A fase de bolha ocorre depois das fases de captura e destino.

Fase de captura A parte do fluxo de evento em que um evento se propaga para baixo, do destino mais geral até o objeto de destino mais específico. A fase de captura ocorre antes das fases de destino e bolhas.

Comportamento padrão Alguns eventos incluem um comportamento que normalmente ocorre ao longo do evento e é conhecido como comportamento padrão. Por exemplo, quando um usuário digita em um campo de texto, ocorre um evento de entrada de texto. O comportamento padrão desse evento é exibir o caractere que realmente foi digitado no campo de texto, mas você pode substituí-lo (se, por algum motivo, não desejar exibir o caractere digitado).

Envio Para notificar o evento ocorrido para os ouvintes de evento.

Evento Algo que acontece em um objeto e que pode ser informado para outros objetos.

Fluxo de evento Quando os eventos acontecem em um objeto na lista de exibição (um objeto exibido na tela), todos os objetos que contêm o objeto em questão são informados sobre o evento e notificam seus ouvintes. Esse processo começa com o palco e continua na lista de exibição até o objeto real onde ocorreu o evento e, em seguida, retorna ao palco. Esse processo também é conhecido como fluxo de evento.

Objeto de evento Um objeto que contém informações sobre a ocorrência de um evento específico, que é enviado para todos os ouvintes assim que o evento acontece.

Destino do evento O objeto que realmente envia um evento. Por exemplo, se o usuário clica em um botão que está dentro de uma entidade gráfica contida no palco, todos esses objetos despacham eventos, mas o destino do evento é o local onde o evento realmente aconteceu – nesse caso, o botão clicado.

Ouvinte Um objeto ou uma função que foi registrada com um objeto para indicar que deve ser notificado quando ocorrer um evento específico.

Fase de destino O ponto do fluxo de evento no qual um evento atingiu o destino mais específico possível. A fase de destino ocorre entre as fases de captura e bolhas.

Como a manipulação de eventos do ActionScript 3.0 difere das versões anteriores

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A diferença mais notável entre a manipulação de eventos no ActionScript 3.0 e nas versões anteriores do ActionScript é o fato de que, no ActionScript 3.0, existe apenas um sistema para manipulação de eventos, enquanto nas versões anteriores existem diversos sistemas diferentes de manipulação de eventos. Esta seção começa com uma visão geral de como era a manipulação de eventos nas versões anteriores do ActionScript e, em seguida, discute como a manipulação de eventos foi alterada para o ActionScript 3.0.

Manipulação de eventos nas versões anteriores do ActionScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As versões anteriores ao ActionScript 3.0 forneciam diversas maneiras diferentes para manipular eventos:

- Manipuladores de eventos `on()` que podem ser colocados diretamente nas ocorrências de Button e de MovieClip
- Manipuladores `onClipEvent()` que podem ser colocados diretamente nas ocorrências de MovieClip
- Propriedades de função de retorno de chamada, como `XML.onload` e `Camera.onActivity`
- Ouvintes de evento registrados ao usar o método `addListener()`
- A classe UIEventDispatcher que implementou parcialmente o modelo de evento DOM.

Cada um desses mecanismos tem vantagens e desvantagens. Os manipuladores `on()` e `onClipEvent()` são fáceis de usar, mas dificultam a manutenção posterior dos projetos porque o código colocado diretamente nos botões e clipes de filme pode ser difícil de localizar. As funções de retorno de chamada também são simples de implementar, mas você só pode usar uma função por evento. Os ouvintes de evento são mais difíceis de implementar porque requerem não só a criação de um objeto de ouvinte e de uma função, mas também o registro do ouvinte com o objeto que gera o evento. No entanto, esse aumento da sobrecarga permite criar vários objetos de ouvinte e registrar todos eles para o mesmo evento.

O desenvolvimento de componentes para o ActionScript 2.0 gerou mais um modelo de evento. Esse novo modelo, incorporado na classe UIEventDispatcher, foi baseado em um subconjunto da especificação de eventos DOM. Os desenvolvedores familiarizados com a manipulação de eventos de componente acharão a transição para o modelo de evento do ActionScript 3.0 relativamente fácil.

Infelizmente, a sintaxe usada pelos diversos modelos de evento é diferente em alguns pontos. Por exemplo, no ActionScript 2.0, algumas propriedades, como `TextField.onChanged`, podem ser usadas como uma função de retorno de chamada ou um ouvinte de evento. No entanto, a sintaxe para registrar objetos de ouvinte é diferente, dependendo do uso de uma das seis classes que oferecem suporte aos ouvintes ou da classe UIEventDispatcher. Para as classes Key, Mouse, MovieClipLoader, Selection, Stage e TextField, use o método `addListener()`, mas, para a manipulação de eventos de componentes, use o método chamado `addEventListener()`.

Outra complexidade decorrente dos diferentes modelos de manipulação de eventos é o escopo da função do manipulador de eventos, que variava muito dependendo do mecanismo usado. Em outras palavras, o significado da palavra-chave `this` não era consistente entre os sistemas de manipulação de eventos.

Manipulação de eventos no ActionScript 3.0

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 apresenta um único modelo de manipulação de eventos que substitui os diversos mecanismos diferentes que existiam nas versões anteriores da linguagem. O novo modelo de evento baseia-se na especificação de eventos DOM (Document Object Model) de nível 3. Embora o formato de arquivo SWF não estejam em conformidade especificamente com o padrão DOM, existem similaridades suficientes entre a lista de exibição e a estrutura do DOM que permitem a implementação do modelo de evento DOM. Um objeto na lista de exibição é equivalente a um nó na estrutura hierárquica DOM e os termos *objeto da lista de exibição* e *nó* são usados alternadamente nesta discussão.

A implementação do Flash Player e do AIR do modelo de evento DOM inclui um conceito chamado comportamento padrão. Um *comportamento padrão* é uma ação executada pelo Flash Player ou AIR como consequência normal de determinados eventos.

Comportamentos padrão

Os desenvolvedores são geralmente responsáveis por gravar o código que responde a eventos. No entanto, em alguns casos, como um comportamento normalmente é associado a um evento, o Flash Player ou o AIR o executa automaticamente a não ser que o desenvolvedor adicione algum código para cancelá-lo. Como o Flash Player ou o AIR exibe o comportamento automaticamente, tais comportamentos são chamados de padrão.

Por exemplo, quando um usuário insere um texto em um objeto TextField, a expectativa de que o texto será exibido nesse objeto TextField é tão comum que o comportamento é incorporado no Flash Player e no AIR. Se não desejar que esse comportamento padrão ocorra, cancele-o usando o novo sistema de manipulação de eventos. Quando o usuário insere o texto em um objeto TextField, o Flash Player ou o AIR cria uma ocorrência da classe TextEvent para representar essa entrada do usuário. Para impedir que o Flash Player ou o AIR exiba o texto no objeto TextField, acesse essa ocorrência específica de TextEvent e chame o método `preventDefault()` dessa ocorrência.

Nem todos os comportamentos padrão podem ser impedidos. Por exemplo, o Flash Player e o AIR geram um objeto MouseEvent quando o usuário clica duas vezes em uma palavra em um objeto TextField. O comportamento padrão, que não pode ser evitado, é a palavra realçada quando o cursor passa.

Vários tipos de objetos de evento não têm comportamentos padrão associados. Por exemplo, o Flash Player envia um objeto de evento connect quando uma conexão de rede é estabelecida, mas não há nenhum comportamento padrão associado a esse objeto. A documentação API da classe Event e das respectivas subclasses lista cada tipo de evento e descreve todo comportamento padrão associado e se esse comportamento pode ser impedido.

É importante mencionar que os comportamentos padrão são associados apenas aos objetos de evento enviados pelo Flash Player ou AIR, e não existem para objetos de evento enviados de modo programático pelo ActionScript. Por exemplo, você pode usar os métodos da classe EventDispatcher para enviar um objeto de evento do tipo `textInput`, mas esse objeto não terá nenhum comportamento padrão associado. Em outras palavras, o Flash Player e o AIR não exibirão um caractere em um objeto TextField em decorrência de um evento `textInput` enviado programaticamente.

Novidades dos ouvintes de evento do ActionScript 3.0

Para desenvolvedores familiarizados com o método `addListener()` do ActionScript 2.0, talvez seja útil descrever as diferenças entre o modelo de ouvinte de evento do ActionScript 2.0 e o do ActionScript 3.0. A lista a seguir descreve as principais diferenças entre os dois modelos de evento:

- Para incluir ouvintes de evento no ActionScript 2.0, você usa `addListener()` em alguns casos e `addEventListener()` em outros, enquanto no ActionScript 3.0, `addEventListener()` é usado em todas as situações.
- Não existe nenhum fluxo de evento no ActionScript 2.0, ou seja, o método `addListener()` pode ser chamado somente no objeto que transmite o evento. Já no ActionScript 3.0, o método `addEventListener()` pode ser chamado em qualquer objeto que faça parte do fluxo de evento.
- No ActionScript 2.0, os ouvintes de evento podem ser funções, métodos ou objetos, enquanto que no ActionScript 3.0, apenas funções ou métodos podem ser ouvintes de evento.

O fluxo de eventos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

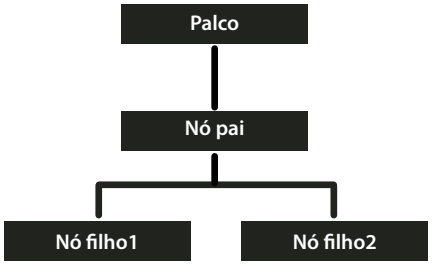
O Flash Player ou o AIR envia objetos de evento sempre que ocorre um evento. Se o destino do evento não estiver na lista de exibição, o Flash Player ou o AIR enviará o objeto diretamente para o destino. Por exemplo, o Flash Player envia o objeto de evento progress diretamente para um objeto URLStream. No entanto, se o destino do evento estiver na lista de exibição, o Flash Player enviará o objeto para a lista de exibição e esse objeto irá percorrer a lista até chegar ao destino.

O *fluxo de evento* descreve como um objeto de evento se move pela lista de exibição. A lista de exibição está organizada em uma hierarquia que pode ser descrita como uma árvore. No topo da hierarquia da lista de exibição está o palco, que é um contêiner especial de objeto de exibição que serve como raiz da lista de exibição. O palco é representado pela classe flash.display.Stage e só pode ser acessado por meio de um objeto de exibição. Cada objeto de exibição tem uma propriedade chamada `stage` que faz referência ao palco desse aplicativo.

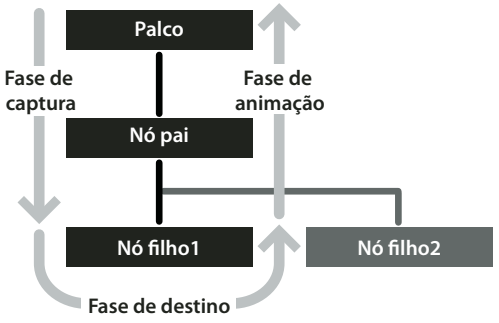
Quando o Flash Player ou o AIR envia um objeto para um evento relacionado à lista de exibição, esse objeto de evento faz uma viagem de ida e volta do palco ao *nó de destino*. A especificação de eventos DOM define o nó de destino como o nó que representa o destino do evento. Em outras palavras, o nó de destino é o objeto da lista de exibição onde ocorreu o evento. Por exemplo, se o usuário clicar em um objeto da lista de exibição chamado `child1`, o Flash Player ou o AIR enviará um objeto de evento usando `child1` como nó de destino.

O fluxo de evento é dividido conceitualmente em três partes. A primeira parte é chamada de fase de captura; essa fase compreende todos os nós do palco ao pai do nó de destino. A segunda parte é chamada de fase de destino e consiste apenas no nó de destino. A terceira parte é chamada de fase de bolhas. A fase de bolhas é composta pelos nós encontrados na viagem de retorno do pai do nó de destino ao palco.

Os nomes das fases farão mais sentido se você pensar na lista de exibição como uma hierarquia vertical com o palco no topo, como mostra o diagrama a seguir:



Se o usuário clicar no `nó Child1`, o Flash Player ou o AIR enviará um objeto ao fluxo de evento. Como mostra a imagem a seguir, a jornada do objeto começa no `palco`, vai até o `nó pai`, segue para o `nó Child1` e volta ao `palco`, movendo-se pelo `nó pai` novamente até voltar ao `palco`.



Neste exemplo, a fase de captura inclui o `palco` e o `nó pai` durante a viagem de ida inicial. A fase de destino consiste no tempo gasto no `nó Child1`. A fase de bolhas inclui o `nó pai` e o `palco`, pois ambos estão presentes na viagem de volta ao nó raiz.

O fluxo de evento contribui com um sistema de manipulação de eventos mais avançado do que o anteriormente disponível para os programadores no ActionScript. Nas versões anteriores do ActionScript, não existe o fluxo de evento, ou seja, os ouvintes de evento podem ser adicionados somente ao objeto que gera o evento. No ActionScript 3.0, é possível adicionar ouvintes de evento não só a um nó de destino, mas também a qualquer nó ao longo do fluxo de evento.

A possibilidade de adicionar ouvintes ao longo do fluxo de evento é útil quando um componente da interface do usuário tem mais de um objeto. Por exemplo, um objeto de botão normalmente contém um objeto de texto que serve como rótulo do botão. Sem poder adicionar um ouvinte ao fluxo de evento, seria necessário adicionar um ouvinte ao objeto de botão e ao objeto de texto para garantir o recebimento de notificações sobre eventos de clique que ocorrem em qualquer lugar no botão. No entanto, a existência do fluxo de evento permite colocar um único ouvinte no objeto de botão que manipula eventos de clique ocorridos no objeto de texto ou nas áreas do objeto de botão que não são obscurecidas pelo objeto de texto.

Porém, nem todos os objetos de evento participam das três fases do fluxo de evento. Alguns tipos de eventos, como `enterFrame` e `init`, são enviados diretamente para o nó de destino e não participam da fase de captura, nem da fase de bolhas. Outros eventos podem ser direcionados para objetos que não estão na lista de exibição, como eventos enviados para uma ocorrência da classe Socket. Esses objetos de evento também vão diretamente para o objeto de destino, sem participar das fases de captura e de bolhas.

Para descobrir como se comporta um tipo de evento específico, consulte a documentação da API ou examine as propriedades do objeto de evento. O exame das propriedades do objeto de evento está descrito na próxima seção.

Objetos de evento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os objetos de evento têm duas finalidades principais no novo sistema de manipulação de eventos. Primeiro, objetos de evento representam eventos reais, armazenando informações sobre eventos específicos em um conjunto de propriedades. Segundo, objetos de evento contêm um conjunto de métodos que permite tratar objetos de evento e afetar o comportamento do sistema de tratamento de eventos.

Para facilitar o acesso a esses métodos e propriedades, a API do Flash Player define uma classe Event que serve como base para todos os objetos de evento. A classe Event define um conjunto fundamental de propriedades e métodos comuns a todos os objetos de evento.

Esta seção começa com uma discussão sobre as propriedades da classe Event, continua com uma descrição dos métodos da classe Event e termina explicando por que existem subclasses de Event.

Noções básicas das propriedades da classe Event

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Event define diversas propriedades e constantes somente leitura que fornecem informações importantes sobre um objeto de evento. As seguintes propriedades são especialmente importantes:

- Os tipos de objeto de evento são representados por constantes e armazenados na propriedade `Event.type`.
- Se for necessário impedir o comportamento padrão de um evento, isso será representado por um valor booliano e armazenado na propriedade `Event.cancelable`.
- As informações de fluxo de evento são contidas nas propriedades restantes.

Tipos de objetos de evento

Todo objeto de evento tem um tipo de evento associado. Os tipos de evento são armazenados na propriedade `Event.type` como valores de sequência. Isso é útil para saber o tipo de objeto de evento, de modo que seu código possa distinguir entre objetos de tipos diferentes. Por exemplo, o código a seguir especifica que a função do ouvinte `clickHandler()` deve responder a qualquer objeto de evento de clique de mouse transmitido para `myDisplayObject`:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

Vinte e quatro tipos de evento são associados à classe Event propriamente dita e representados por constantes da classe Event; alguns deles são mostrados no trecho a seguir da definição da classe Event:

```
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
    }
}
```

Essas constantes facilitam a referência a tipos de evento específicos. Use essas constantes em vez das strings que representam. Se você digitar o nome de uma constante incorretamente no código, o compilador detectará o erro, mas, se as strings forem utilizadas, um erro tipográfico talvez não se manifeste no tempo de compilação e gere um comportamento inesperado difícil de depurar. Por exemplo, ao adicionar um ouvinte de evento, use o seguinte código:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

em vez de:

```
myDisplayObject.addEventListener("click", clickHandler);
```

Informações de comportamento padrão

Seu código pode verificar se o comportamento padrão de um objeto de evento determinado pode ser impedido, acessando a propriedade `cancelable`. A propriedade `cancelable` tem um valor booliano que indica se um comportamento padrão pode ou não ser impedido. Você pode impedir ou cancelar o comportamento padrão associado a um pequeno número de eventos, usando o método `preventDefault()`. Para obter mais informações, consulte Cancelamento do comportamento padrão do evento em "Noções básicas dos métodos da classe Event" na página 134.

Informações sobre o fluxo de evento

As demais propriedades da classe Event contêm informações importantes sobre um objeto de evento e sua relação com o fluxo de evento, conforme descrito na lista a seguir:

- A propriedade `bubbles` contém informações sobre as partes do fluxo nas quais o objeto de evento participa.
- A propriedade `eventPhase` indica a fase atual no fluxo de evento.
- A propriedade `target` armazena uma referência ao destino do evento.
- A propriedade `currentTarget` armazena uma referência ao objeto da lista de exibição que está processando o objeto de evento no momento.

A propriedade bubbles

Um evento é animado se seu objeto participa da fase de bolhas do fluxo de evento, o que significa que o objeto de evento volta do nó de destino para seu ancestral até chegar ao palco. A propriedade `Event.bubbles` armazena um valor booliano que indica se o objeto de evento participa na fase de bolhas. Como todos os eventos que são animados também participam nas fases de captura e de destino, qualquer evento animado participa das três fases do fluxo de evento. Se o valor for `true`, o objeto de evento participará das três fases. Se o valor for `false`, o objeto de evento não participará na fase de bolhas.

A propriedade eventPhase

Você pode determinar a fase de qualquer objeto de evento investigando a propriedade `eventPhase`. A propriedade `eventPhase` contém um valor inteiro sem sinal que representa uma das três fases do fluxo de evento. A API do Flash Player define uma classe EventPhase separada que contém três constantes que correspondem a três valores inteiros sem sinal, como mostra o seguinte trecho de código:

```
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;
    }
}
```

Essa constantes correspondem a três valores válidos da propriedade `eventPhase`. Você pode usar essas constantes para deixar seu código mais legível. Por exemplo, se desejar assegurar que uma função `myFunc()` seja chamada somente se o destino do evento estiver no palco de destino, use o código a seguir para testar essa condição:

```
if (event.eventPhase == EventPhase.AT_TARGET)
{
    myFunc();
}
```

A propriedade target

A propriedade `target` armazena uma referência ao objeto que é o destino do evento. Em alguns casos, isso é simples, como quando um microfone é ativado; o destino do objeto de evento é o objeto Microphone. No entanto, se o destino estiver na lista de exibição, a hierarquia da lista deve ser levada em consideração. Por exemplo, se o usuário inserir um clique de mouse em um ponto que inclui objetos sobrepostos da lista de exibição, o Flash Player e o AIR sempre escolherão o objeto mais distante do palco como destino do evento.

Para arquivos SWF complexos, especialmente aqueles nos quais os botões são decorados regularmente com objetos filho menores, a propriedade `target` talvez não seja usada com frequência porque, em geral, apontará para o objeto filho de um botão, não para o botão. Nesses casos, a prática comum é adicionar ouvintes de evento ao botão e usar a propriedade `currentTarget` que aponta para o botão, já que a propriedade `target` pode apontar para um filho do botão.

A propriedade currentTarget

A propriedade `currentTarget` contém uma referência ao objeto que está processando o objeto de evento no momento. Embora possa parecer estranho não saber qual nó está processando o objeto de evento que você está examinando no momento, é possível adicionar uma função de ouvinte a qualquer objeto de exibição do fluxo desse objeto de evento; a função de ouvinte pode ser colocada em qualquer lugar. Além disso, a mesma função de ouvinte pode ser adicionada a diferentes objetos de exibição. À medida que aumenta o tamanho e a complexidade de um projeto, a propriedade `currentTarget` fica cada vez mais útil.

Noções básicas dos métodos da classe Event

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Há três categorias de métodos da classe Event:

- Métodos utilitários, que podem criar cópias de um objeto de evento ou convertê-lo em uma string

- Métodos de fluxo de evento, que removem objetos do fluxo de evento
- Métodos de comportamento padrão, que impedem o comportamento padrão ou verificam se ele foi impedido

Métodos utilitários da classe Event

Existem dois métodos de utilitário na classe Event. O método `clone()` permite criar cópias de um objeto de evento. O método `toString()` permite gerar uma representação de string das propriedades de um objeto de evento juntamente com os respectivos valores. Esses dois métodos são usados internamente pelo sistema de modelo de evento, mais são expostos aos desenvolvedores para uso geral.

Para desenvolvedores avançados que criam subclasses de Event, é necessário substituir e implementar versões dos dois métodos de utilitário para garantir que a subclasse funcione corretamente.

Interrupção do fluxo de evento

Você pode chamar o método `Event.stopPropagation()` ou o método `Event.stopImmediatePropagation()` para impedir que um objeto de evento continue seu percurso no fluxo. Os dois métodos são praticamente idênticos e diferem apenas quanto aos ouvintes de evento do nó atual que podem ser executados:

- O método `Event.stopPropagation()` impede que o objeto de evento se mova para o próximo nó, mas só depois que algum outro ouvinte de evento do nó atual tenha permissão para ser executado.
- O método `Event.stopImmediatePropagation()` também impede que o objeto de evento se mova para o próximo nó, mas não permite que outros ouvintes de evento do nó atual sejam executados.

Chamar um desses métodos não afeta a ocorrência do comportamento padrão associado a um evento. Use os métodos do comportamento padrão da classe Event para impedir tal comportamento.

Cancelamento de comportamento de evento padrão

Os dois métodos pertinentes ao cancelamento de comportamento padrão são os métodos `preventDefault()` e `isDefaultPrevented()`. Chame o método`preventDefault()` para cancelar o comportamento padrão associado ao evento. Para verificar se `preventDefault()` já foi chamado em um objeto de evento, chame o método `isDefaultPrevented()`, que retorna o valor `true` se o método já tiver sido chamado; caso contrário, retornará `false`.

O método `preventDefault()` só funcionará se o comportamento padrão do evento puder ser cancelado. Para verificar isso, consulte a documentação da API para esse tipo de evento ou use o ActionScript para examinar a propriedade `cancelable` do objeto de evento.

Cancelar o comportamento padrão não tem nenhum efeito no andamento de um objeto de evento através do fluxo de evento. Use os métodos de fluxo de evento da classe Event para remover um objeto de evento do fluxo de evento.

Subclasses da classe Event

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Em muitos eventos, o conjunto comum de propriedades definido na classe Event é suficiente. No entanto, outros eventos têm características exclusivas que não podem ser capturadas pelas propriedades disponíveis na classe Event. Para esses eventos, o ActionScript 3.0 define várias subclasses da classe Event.

Cada subclasse oferece propriedades e tipos de eventos adicionais exclusivos a essa categoria de eventos. Por exemplo, os eventos relacionados à entrada do mouse têm várias características exclusivas que não podem ser capturadas pelas propriedades definidas na classe Event. A classe MouseEvent estende a classe Event adicionando dez propriedades que contêm informações como o local do evento de mouse e que indicam se teclas específicas foram pressionadas durante o evento de mouse.

Uma subclasse de Event também contém constantes que representam os tipos de evento associados à subclasse. Por exemplo, a classe MouseEvent define constantes para vários tipos de evento de mouse, incluindo `click`, `doubleClick`, `mouseDown` e `mouseUp`.

Conforme descrito na seção de métodos de utilitário da classe Event em"Objetos de evento" na página 132, é preciso substituir os métodos `clone()` e `toString()` para oferecer funcionalidade específica para a subclasse.

Ouvintes de evento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os ouvintes de evento, também chamados de manipuladores de evento, são funções executadas pelo Flash Player e pelo AIR em resposta a eventos específicos. A adição de um ouvinte de evento é um processo de duas etapas. Primeiro, crie um método de função ou classe a ser executado pelo Flash Player ou AIR em resposta ao evento. Às vezes, isso é chamado de função de ouvinte ou de manipulador de evento. Segundo, use o método `addEventListener()` para registrar a função de ouvinte no destino do evento ou em qualquer objeto da lista de exibição ao longo do fluxo de evento adequado.

Criação de uma função de ouvinte

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A criação de funções de ouvinte é uma das diferenças entre o modelo de evento do ActionScript 3.0 e o modelo de evento DOM. No modelo de evento DOM, existe uma distinção clara entre um ouvinte de evento e uma função de ouvinte: um ouvinte de evento é uma ocorrência de uma classe que implementa a interface EventListener, enquanto a função de ouvinte é um método dessa classe chamado `handleEvent()`. No modelo de evento DOM, a ocorrência da classe que contém a função de ouvinte é registrada, não a função de ouvinte propriamente dita.

No modelo de evento do ActionScript 3.0, não existe diferença entre um ouvinte de evento e uma função de ouvinte. O ActionScript 3.0 não tem uma interface EventListener e as funções de ouvinte podem ser definidas fora de uma classe ou como parte dela. Além disso, as funções de ouvinte não precisam ser chamadas de `handleEvent()` - elas podem ser chamadas com qualquer identificador válido. No ActionScript 3.0, o nome da função de ouvinte real é registrado.

Função de ouvinte definida fora de uma classe

O código a seguir cria um arquivo SWF simples que exibe um quadrado vermelho. Uma função de ouvinte chamada `clickHandler()`, que não faz parte de uma classe, ouve os eventos de clique de mouse no quadrado vermelho.

```
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);
}
```

Quando um usuário interage com o arquivo SWF resultante clicando no quadrado, o Flash Player ou AIR gera a seguinte saída de traço:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Observe que o objeto de evento é transmitido como um argumento para `clickHandler()`. Isso permite que a função de ouvinte examine o objeto de evento. Neste exemplo, use a propriedade `type` do objeto de evento para certificar-se de que o evento é um evento de clique.

O exemplo também verifica o valor da palavra-chave `this`. Nesse caso, `this` representa o objeto global, o que faz sentido porque a função é definida fora de um objeto ou classe personalizado.

Função de ouvinte definida como um método de classe

O exemplo a seguir é idêntico ao anterior que define a classe ClickExample, mas a função `clickHandler()` é definida como um método da classe ChildSprite:

```
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);
    }
}
```

Quando um usuário interage com o arquivo SWF resultante clicando no quadrado vermelho, o Flash Player ou AIR gera a seguinte saída de traço:

```
clickHandler detected an event of type: click
the this keyword refers to: [object ChildSprite]
```

Observe que a palavra-chave `this` faz referência à ocorrência de ChildSprite chamada `child`. Este comportamento é diferente do ActionScript 2.0. Caso já tenha usado componentes no ActionScript 2.0, talvez se lembre que, quando um método de classe era transmitido para `UIEventDispatcher.addEventListener()`, o escopo do método ia até o componente que transmite o evento em vez da classe na qual o método de ouvinte era definido. Em outras palavras, se você utilizou essa técnica no ActionScript 2.0, a palavra-chave `this` fazia referência ao componente que transmite o evento, não à ocorrência de ChildSprite.

Isso foi um problema significativo para alguns programadores, pois não permitia o acesso a outros métodos e propriedades da classe que contém o método de ouvinte. Como solução temporária, os programadores do ActionScript 2.0 podiam usar a classe `mx.util.Delegate` para alterar o escopo do método de ouvinte. No entanto, isso não é mais necessário porque o ActionScript 3.0 cria um método vinculado quando `addEventListener()` é chamado. Em resultado disso, a palavra-chave `this` faz referência à ocorrência de ChildSprite chamada `child`, e o programador tem acesso a outros métodos e propriedades da classe ChildSprite.

Ouvinte de evento que não deve ser usado

Existe uma terceira técnica que permite criar um objeto genérico com uma propriedade que aponta para uma função de ouvinte atribuída dinamicamente, mas essa técnica não é recomendada. Ela será discutida aqui porque foi muito usada no ActionScript 2.0, mas não deve ser usada no ActionScript 3.0. Essa técnica não é recomendada porque a palavra-chave `this` fará referência ao objeto global em vez do objeto de ouvinte.

O exemplo a seguir é idêntico ao exemplo anterior da classe ClickExample, mas a função de ouvinte é definida como parte de um objeto genérico chamado `myListenerObj`:

```
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);
}
```

Os resultados do traço serão similares a:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Você talvez pensasse que `this` faria referência a `myListenerObj` e que a saída de traço seria `[object Object]`, mas a referência foi feita ao objeto global. Ao transmitir o nome de uma propriedade dinâmica como um argumento para `addEventListener()`, o Flash Player ou AIR não consegue criar um método vinculado. Isso ocorre porque, quando está transmitindo o parâmetro `listener`, você não está transmitindo nada mais do que o endereço de memória da função de ouvinte, e o Flash Player e AIR não conseguem vincular esse endereço de memória com a ocorrência de `myListenerObj`.

Gerenciamento de ouvintes de evento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode gerenciar suas funções de ouvinte usando os métodos da interface IEventDispatcher. A interface IEventDispatcher é a versão ActionScript 3.0 da interface EventTarget do modelo de evento DOM. Embora o nome IEventDispatcher possa transmitir a ideia de que o objetivo principal é enviar (ou despachar) objetos de evento, os métodos dessa classe são usados com mais frequência para registrar, verificar e remover ouvintes de evento. A interface IEventDispatcher define cinco métodos, como mostra o código a seguir:

```
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;
    }
}
```

A API do Flash Player implementa a interface IEventDispatcher com a classe EventDispatcher, que serve como base para todas as classes que podem ser destinos de evento ou fazer parte de um fluxo de evento. Por exemplo, a classe DisplayObject é herdada da classe EventDispatcher. Isso significa que todos os objetos da lista de exibição têm acesso aos métodos da interface IEventDispatcher.

Adição de ouvintes de evento

O método `addEventListener()` é a força motriz da interface IEventDispatcher. Você pode usá-lo para registrar suas funções de ouvinte. Os dois parâmetros necessários são `type` e `listener`. Use o parâmetro `type` para especificar o tipo de evento. Use o parâmetro `listener` para especificar a função de ouvinte que será executada quando o evento ocorrer. O parâmetro `listener` pode ser uma referência a um método de classe ou função.

Não use parênteses quando especificar o parâmetro `listener`. Por exemplo, a função `clickHandler()` é especificada sem parênteses na seguinte chamada do método `addEventListener()`:

```
addEventListener(MouseEvent.CLICK, clickHandler)
```

O parâmetro `useCapture` do método `addEventListener()` permite controlar a fase do fluxo de evento na qual o ouvinte estará ativo. Se `useCapture` for definido como `true`, o ouvinte estará ativo durante a fase de captura do fluxo de evento. Se `useCapture` for definido como `false`, o ouvinte estará ativo durante as fases de destinos e de bolhas do fluxo de evento. Para ouvir um evento durante todas as fases do fluxo de evento, chame `addEventListener()` duas vezes, uma com `useCapture` definido como `true` e outra com `useCapture` definido como `false`.

O parâmetro `priority` do método `addEventListener()` não é uma parte oficial do modelo de evento DOM nível 3. Ele está incluído no ActionScript 3.0 para fornecer mais flexibilidade para organizar os ouvintes de evento. Ao chamar `addEventListener()`, você pode definir a prioridade desse ouvinte de evento transmitindo um valor inteiro como o parâmetro `priority`. O valor padrão é 0, mas é possível definir valores inteiros negativos ou positivos. Quanto maior o número, mais rapidamente o ouvinte de evento será executado. Os ouvintes de evento com a mesma prioridade são executados na ordem em que foram adicionados, de modo que o ouvinte adicionado primeiro será executado antes.

O parâmetro `useWeakReference` permite especificar se a referência à função de ouvinte será fraca ou normal. Se esse parâmetro for definido como `true`, você poderá evitar situações nas quais as funções de ouvinte persistem na memória mesmo quando não são mais necessárias. O Flash Player e o AIR usam uma técnica chamada *coleta de lixo* para apagar da memória os objetos que não são mais usados. Um objeto não é mais necessário quando não existe nenhuma referência a ele. O coletor de lixo ignora as referências fracas e, desse modo, uma função de ouvinte que tem apenas uma referência fraca se qualifica para a coleta de lixo.

Remoção de ouvintes de eventos

Você pode usar o método `removeEventListener()` para remover um ouvinte de evento que não seja mais necessário. É uma boa ideia remover o ouvinte que não será mais usado. Entre os parâmetros obrigatórios estão `eventName` e `listener`, que são os mesmos parâmetros obrigatórios no método `addEventListener()`. Não se esqueça que é possível ouvir eventos durante todas as fases chamando `addEventListener()` duas vezes, uma com `useCapture` definido como `true` e outra definido como `false`. Para remover os dois ouvintes de evento, seria necessário chamar `removeEventListener()` duas vezes, uma com `useCapture` definido como `true` e outra definido como `false`.

Envio de eventos

O método `dispatchEvent()` pode ser usado por programadores avançados para enviar um objeto de evento personalizado para o fluxo de evento. O único parâmetro aceito por esse método é uma referência a um objeto de evento, que deve ser uma ocorrência da classe Event ou uma subclasse de Event. Assim que é enviada, a propriedade `target` do objeto de evento é definida como o objeto no qual `dispatchEvent()` foi chamado.

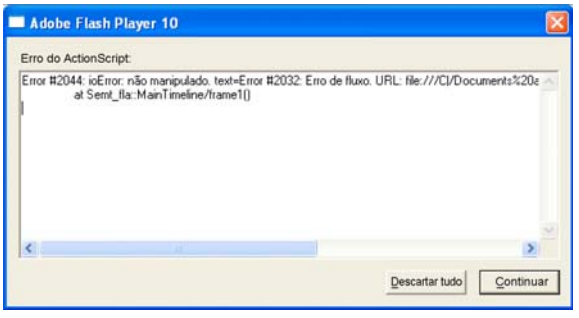
Verificação da existência de ouvintes de eventos

Os dois métodos finais da interface IEventDispatcher fornecem informações úteis sobre a existência de ouvintes de evento. O método `hasEventListener()` retornará `true` se um ouvinte de evento for encontrado para um tipo de evento específico em um determinado objeto da lista de exibição. O método `willTrigger()` também retornará `true` se um ouvinte for encontrado para um objeto específico da lista de exibição, mas `willTrigger()` procurará ouvintes nesse objeto de exibição e em todos os ancestrais desse objeto em todas as fases do fluxo de evento.

Eventos de erro sem ouvintes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As exceções, em vez dos eventos, são o principal mecanismo de manipulação de erros no ActionScript 3.0, mas a manipulação de exceções não funciona para operações assíncronas, como o carregamento de arquivos. Se ocorrer um erro durante uma operação assíncrona, o Flash Player e o AIR enviarão um objeto de evento de erro. Se nenhum ouvinte for criado para o evento de erro, as versões do depurador do Flash Player e do AIR exibirão uma caixa de diálogo com informações sobre o erro. Por exemplo, a versão do depurador do Flash Player exibe a caixa de diálogo a seguir, que descreve o erro quando o aplicativo tentar carregar um arquivo a partir de URLs inválidas:



A maioria dos eventos de erro baseia-se na classe ErrorEvent e, desse modo, tem uma propriedade chamada `text` que é usada para armazenar a mensagem de erro exibida pelo Flash Player ou AIR. As duas exceções são as classes StatusEvent e NetStatusEvent. Essas duas classes têm uma propriedade `level` (`StatusEvent.level` e `NetStatusEvent.info.level`). Quando o valor da propriedade `level` é "`error`", esses tipos de evento são considerados eventos de erro.

Um evento de erro não interrompe a execução do arquivo SWF. Ele será manifestado somente como uma caixa de diálogo nas versões do depurador dos plug-ins do navegador e de players dedicados, como uma mensagem no painel de saída do player de criação e como uma entrada no arquivo de log do Adobe Flash Builder. Ele não se manifestará de modo algum nas versões do Flash Player ou do AIR.

Exemplo de tratamento de evento: relógio de alarme

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo do Alarm Clock consiste em um relógio que permite que o usuário especifique um horário no qual o alarme deve ser desativado, bem como uma mensagem a ser exibida nesse horário. O exemplo do Alarm Clock baseia-se no aplicativo SimpleClock da seção "Trabalho com datas e horas" na página 1 e ilustra vários aspectos do trabalho com eventos ActionScript 3.0, incluindo:

- Como ouvir e responder a um evento
- Notificação de um evento aos ouvintes
- Criação de um tipo de evento personalizado

Para obter os arquivos do aplicativo Flash Professional dessa amostra, consulte http://www.adobe.com/go/learn_programmingAS3samples_flash_br. Para obter os arquivos do aplicativo Flex dessa amostra, consulte http://www.adobe.com/go/as3examples_br. Os arquivos do aplicativo Alarm Clock estão localizados na pasta Amostras/AlarmClock. O aplicativo inclui estes arquivos:

Arquivo	Descrição
AlarmClockApp.mxml ou AlarmClockApp.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/clock/AlarmClock.as	Uma classe que estende a classe SimpleClock, adicionando a funcionalidade do despertador.
com/example/programmingas3/clock/AlarmEvent.as	Uma classe de evento personalizada (uma subclasse de flash.events.Event) que serve como objeto do evento `alarm` da classe AlarmClock.
com/example/programmingas3/clock/AnalogClockFace.as	Desenha a superfície de um relógio redondo e os ponteiros de horas, minutos e segundos com base na hora (descrito no exemplo de SimpleClock).
com/example/programmingas3/clock/SimpleClock.as	Um componente da interface do relógio com um recurso simples de marcação da hora (descrito no exemplo de SimpleClock).

Visão geral do Alarm Clock

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O principal recurso do relógio neste exemplo, incluindo o controle da hora e a exibição da superfície do relógio, reutiliza o código do aplicativo SimpleClock, descrito em "Exemplo de data e hora: relógio analógico simples" na página 6. A classe AlarmClock estende a classe SimpleClock desse exemplo, adicionando a funcionalidade necessária para um despertador, incluindo a definição da hora do alarme e do envio da notificação quando o alarme for "desativado".

Os eventos são gerados para enviar a notificação sobre algo ocorrido. A classe AlarmClock expõe o evento Alarm, que outros objetos podem ouvir para executar as ações desejadas. Além disso, a classe AlarmClock usa uma ocorrência da classe Timer para determinar quando o alarme deve ser acionado. Assim como AlarmClock, a classe Timer gera um evento para notificar outros objetos (neste caso, uma ocorrência de AlarmClock) após um determinado período. Assim como a maioria dos aplicativos do ActionScript, os eventos formam uma parte importante da funcionalidade do aplicativo Alarm Clock de exemplo.

Acionamento do alarme

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Conforme mencionado anteriormente, a única funcionalidade da classe AlarmClock está relacionada com a definição e o acionamento do alarme. A classe incorporada Timer (flash.utils.Timer) permite que o desenvolvedor defina o código que será executado após o período especificado. A classe AlarmClock usa uma ocorrência de Timer para determinar quando o alarme deve ser desativado.

```
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);
}
```

A ocorrência de Timer definida na classe AlarmClock chama-se `alarmTimer`. O método `initClock()`, que executa as operações de configuração necessárias para a ocorrência de AlarmClock, faz duas coisas com a variável `alarmTimer`. Primeiro, a variável é percorrida com os parâmetros que instruem a ocorrência de Timer a aguardar 0 milissegundos e acionar o evento timer apenas uma vez. Depois de percorrer `alarmTimer`, o código chama o método `addEventListener()` da variável para indicar se deseja ouvir o evento `timer` dessa variável. Uma ocorrência de Timer funciona enviando o evento `timer` após um período especificado. A classe AlarmClock não precisa saber quando o evento `timer` é enviado para desativar seu próprio alarme. Chamando `addEventListener()`, o código de AlarmClock se autorregistra como ouvinte em `alarmTimer`. Os dois parâmetros indicam que a classe AlarmClock deseja ouvir o evento `timer` (indicado pela constante `TimerEvent.TIMER`) e que, quando o evento ocorrer, o método `onAlarm()` da classe AlarmClock deve ser chamado em resposta a esse evento.

Para definir o alarme realmente, o método `setAlarm()` da classe AlarmClock é chamado do seguinte modo:

```
/**
 * 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;
}
```

Esse método faz várias coisas, como armazenar a mensagem do alarme e criar um objeto Date (`alarmTime`) que representa o momento real em que o alarme é desativado. O mais importante nisto tudo, nas várias linhas finais do método, é o fato de o objeto timer da variável `alarmTimer` ser definido e ativado. Primeiro, o método `reset()` é chamado, interrompendo o cronômetro e redefinindo-o caso já esteja em execução. Em seguida, o horário atual (representado pela variável `now`) é subtraído do valor da variável `alarmTime` para determinar quantos milissegundos devem se passar até o alarme ser desativado. A classe Timer não aciona o evento `timer` em um tempo absoluto, de modo que essa diferença relativa de tempo é atribuída à propriedade `delay` de `alarmTimer`. Finalmente, o método `start()` é chamado para iniciar o cronômetro realmente.

Assim que o período especificado termina, `alarmTimer` envia o evento `timer`. Com a classe AlarmClock registrou o método `onAlarm()` como um ouvinte desse evento, quando o evento `timer` acontece, `onAlarm()` é chamado.

```
/**
 * 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);
}
```

Um método que é registrado como ouvinte de evento deve ser definido com a assinatura apropriada (isto é, o conjunto de parâmetros e o tipo de retorno do método). Para ser ouvinte do evento `timer` da classe Timer, um método deve definir um parâmetro cujo tipo de dados seja TimerEvent (flash.events.TimerEvent), uma subclasse de Event. Quando chama seus ouvintes de evento, a ocorrência de Timer transmite uma ocorrência de TimerEvent como objeto de evento.

Notificação do alarme a outros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Assim como a classe Timer, a classe AlarmClock fornece um evento que permite que outros códigos recebam notificações quando o alarme é desativado. Para que uma classe use a estrutura de manipulação de eventos incorporada no ActionScript, essa classe deve implementar a interface flash.events.IEventDispatcher. Normalmente, isso é feito por meio da extensão da classe flash.events.EventDispatcher, que fornece uma implementação padrão de IEventDispatcher (ou por meio da extensão de uma das subclasses de EventDispatcher). Como descrito anteriormente, a classe AlarmClock estende a classe SimpleClock, que (através de uma cadeia de herança) estende a classe EventDispatcher. Tudo isso significa que a classe AlarmClock já tem uma funcionalidade interna para fornecer seus próprios eventos.

Outro código pode ser registrado para ser notificado sobre o evento `alarm` da classe AlarmClock chamando o método `addEventListener()` que AlarmClock herda de EventDispatcher. Quando uma ocorrência de AlarmClock está pronta para notificar outro código sobre a geração do evento `alarm`, o método `dispatchEvent()`, que também é herdado de EventDispatcher, é chamado.

```
var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
this.dispatchEvent(alarm);
```

Essas linhas de código foram obtidas do método `onAlarm()` da classe AlarmClock (mostrada por completo antes). O método `dispatchEvent()` da ocorrência de AlarmClock é chamado e notifica todos os ouvintes registrados sobre o acionamento do evento `alarm` da ocorrência de AlarmClock. O parâmetro transmitido para `dispatchEvent()` é o objeto de evento que será transmitido junto com os métodos de ouvinte. Nesse caso, é uma ocorrência da classe AlarmEvent, uma subclasse de Event criada especificamente para este exemplo.

Fornecimento de um evento de alarme personalizado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Todos os ouvintes de evento recebem um parâmetro de objeto de evento com informações sobre o evento específico que está sendo acionado. Em muitos casos, o objeto de evento é uma ocorrência da classe Event. No entanto, em alguns casos é útil fornecer informações adicionais aos ouvintes de evento. Uma maneira comum de fazer isso é definir uma nova classe, uma subclasse de Event, e usar uma ocorrência dessa classe como objeto de evento. Neste exemplo, uma ocorrência de AlarmEvent é usada como objeto quando o evento `alarm` da classe AlarmClock é enviado. A classe AlarmEvent, mostrada aqui, fornece informações adicionais sobre o evento `alarm`, especificamente a mensagem de alarme:

```
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;
    }
    ...
}
```

A melhor maneira de criar uma classe de objeto de evento personalizada é definir uma classe que estende a classe Event, como mostrou o exemplo anterior. Para complementar a funcionalidade herdada, a classe AlarmEvent define uma propriedade `message` que contém o texto da mensagem de alarme associada ao evento; o valor de `message` é transmitido como um parâmetro no construtor AlarmEvent. A classe AlarmEvent também define a constante `ALARM`, que pode ser usada para fazer referência ao evento específico (`alarm`) ao chamar o método `addEventListener()` da classe AlarmClock.

Além de adicionar a funcionalidade personalizada, cada subclasse de Event deve substituir o método `clone()` herdado como parte da estrutura de manipulação de eventos do ActionScript. As subclasses de Event também podem substituir o método `toString()` herdado para incluir as propriedades do evento personalizado no valor retornado quando o método `toString()` é chamado.

```
    /**
     * 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");
    }
```

O método `clone()` substituído precisa retornar uma nova ocorrência da subclasse personalizada de Event, com todas as propriedades personalizadas definidas para corresponder à ocorrência atual. No método `toString()` substituído, o método de utilitário `formatToString()` (herdado de Event) é usado para fornecer uma string com o nome do tipo personalizado, bem como nomes e valores de todas as propriedades.

Capítulo 9: Trabalhar com domínios de aplicativo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O objetivo da classe ApplicationDomain é armazenar uma tabela de definições do ActionScript 3.0. Todo código em um arquivo SWF é definido para estar presente em um domínio de aplicativo. Domínios de aplicativo são usados para particionar classes que estão no mesmo domínio de segurança. Eles permitem que haja várias definições da mesma classe e permitem que filhos reutilizem definições de pai.

É possível usar domínios de aplicativo carregando um arquivo SWF externo escrito no ActionScript 3.0 usando a API da classe Loader. (Observe que não é possível usar domínios de aplicativo ao carregar uma imagem ou arquivo SWF no ActionScript 1.0 ou no ActionScript 2.0.) Todas as definições do ActionScript 3.0 contidas na classe carregada são armazenadas no domínio do aplicativo. Ao carregar o arquivo SWF, é possível especificar que o arquivo seja incluído no mesmo domínio do aplicativo que o objeto Loader, definindo o parâmetro `applicationDomain` do objeto LoaderContext como `ApplicationDomain.currentDomain`. Colocando o arquivo SWF carregado no mesmo domínio de aplicativo, é possível acessar suas classes diretamente. Isso poderá ser útil se estiver carregando um arquivo SWF contendo mídia incorporada que permita acesso por meio de nomes de classes associadas ou se você desejar acessar os métodos do arquivo SWF carregado.

O exemplo a seguir parte da premissa de que ele tem acesso para separar o arquivo Greeter.swf que define um método público denominado welcome():

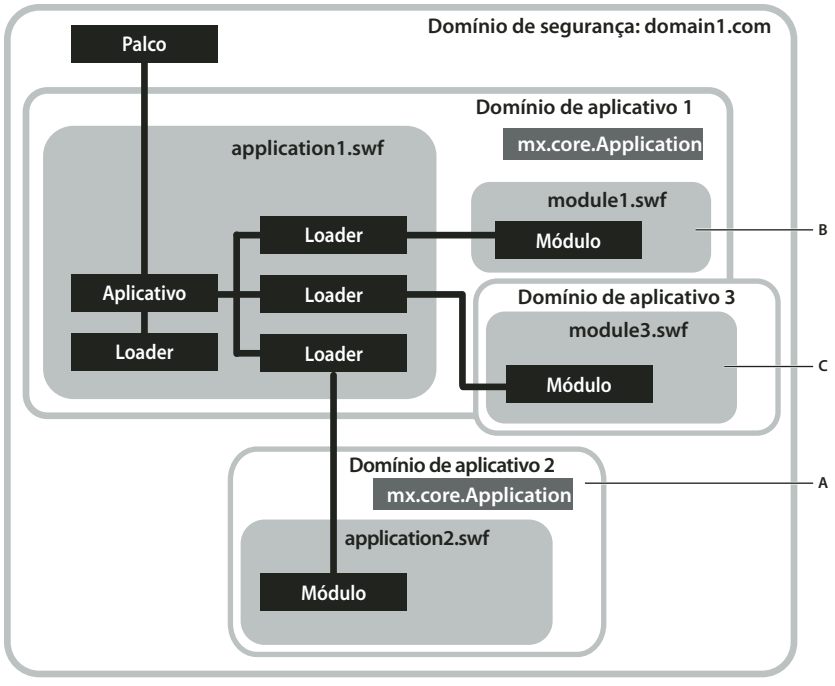
```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLRequest;
    import flash.system.ApplicationDomain;
    import flash.system.LoaderContext;

    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
        }
    }
}
```

Consulte também Exemplo de classe ApplicationDomain da Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Outras coisas que devem ser lembradas ao trabalhar com domínios de aplicativos incluem:

- Todo código em um arquivo SWF é definido para estar presente em um domínio de aplicativo. O *domínio atual* é onde seu aplicativo principal é executado. O *domínio do sistema* contém todos os domínios de aplicativos, incluindo o domínio atual, o que significa que ele contém todas as classes do Flash Player.
- Todos os domínios de aplicativos, exceto o domínio do sistema, têm um domínio-pai associado. O domínio-pai do domínio do aplicativo principal é o domínio do sistema. As classes carregadas só são definidas quando o pai ainda não as definiu. Você não pode substituir uma definição de classe loaded por uma definição mais recente.

O diagrama a seguir mostra um aplicativo que carrega conteúdo de vários arquivos SWF dentro de um único domínio, domain1.com. Dependendo do conteúdo carregado, diferentes domínios de aplicativos podem ser usados. O texto a seguir descreve a lógica usada para definir o domínio apropriado do aplicativo para cada arquivo SWF no aplicativo.



A. *Uso A* ***B.*** *Uso B* ***C.*** *Uso C*

O arquivo do aplicativo principal é application1.swf. Ele contém objetos Loader que carregam conteúdo de outros arquivos SWF. Neste cenário, o domínio atual é o Domínio do aplicativo 1. Uso A, uso B e uso C ilustram diferentes técnicas para configurar o domínio do aplicativo apropriado para cada arquivo SWF em um aplicativo.

Uso A Particionar o arquivo SWF filho criando um filho do domínio do sistema. No diagrama, o Domínio do aplicativo 2 é criado como um filho do domínio do sistema. O arquivo application2.swf é carregado no Domínio do aplicativo 2 e suas definições de classe são portanto particionadas a partir das classes definidas no application1.swf.

Um dos usos dessa técnica é fazer com que um aplicativo antigo carregue uma versão mais nova do mesmo aplicativo sem conflito. Não há conflito porque, embora os mesmos nomes de classes sejam usados, elas são particionadas em diferentes domínios de aplicativo.

O código a seguir cria um domínio de aplicativo que é filho do domínio do sistema, e começa carregando um SWF usando esse domínio de aplicativo:

```
var appDomainA:ApplicationDomain = new ApplicationDomain();

var contextA:LoaderContext = new LoaderContext(false, appDomainA);
var loaderA:Loader = new Loader();
loaderA.load(new URLRequest("application2.swf"), contextA);
```

Uso B: Adicione novas definições de classe às definições de classe atuais. O domínio do aplicativo do module1.swf é definido como o domínio atual (Domínio do aplicativo 1). Isso permite adicionar o conjunto atual de aplicativos das definições de classes com novas definições de classes. Isso pode ser usado para uma biblioteca de tempo de execução do aplicativo principal. O SWF carregado é tratado como uma RSL (biblioteca remota compartilhada). Use essa técnica para carregar RSLs por meio de um pré-carregador antes do início do aplicativo.

O código a seguir carrega um SWF definindo seu domínio de aplicativo para o domínio atual:

```
var appDomainB:ApplicationDomain = ApplicationDomain.currentDomain;

var contextB:LoaderContext = new LoaderContext(false, appDomainB);
var loaderB:Loader = new Loader();
loaderB.load(new URLRequest("module1.swf"), contextB);
```

Uso C: Use as definições de classes pai criando um novo domínio filho do domínio atual. O domínio do aplicativo do module3.swf é filho do domínio atual, e o filho usa as versões de todas as classes do pai. Um uso dessa técnica pode ser um módulo de um RIA (aplicativo de Internet avançado) de várias telas, carregado como um filho do aplicativo principal que usa os tipos do aplicativo principal. Se você puder garantir que todas as classes são sempre atualizadas para serem compatíveis com versões anteriores e que o aplicativo carregado é sempre mais novo do que as coisas que ele carrega, os filhos usarão as versões do pai. Ter um novo domínio de aplicativo também permite descarregar todas as definições de classe para coleta de lixo, se você puder garantir que referências ao SWF filho não continuam a existir.

Essa técnica permite que módulos carregados compartilhem os objetos singleton do carregador e membros da classe estática.

O código a seguir cria um novo domínio filho do domínio atual e inicia o carregamento de um SWF usando aquele domínio de aplicativo:

```
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);
```

Capítulo 10: Programação de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

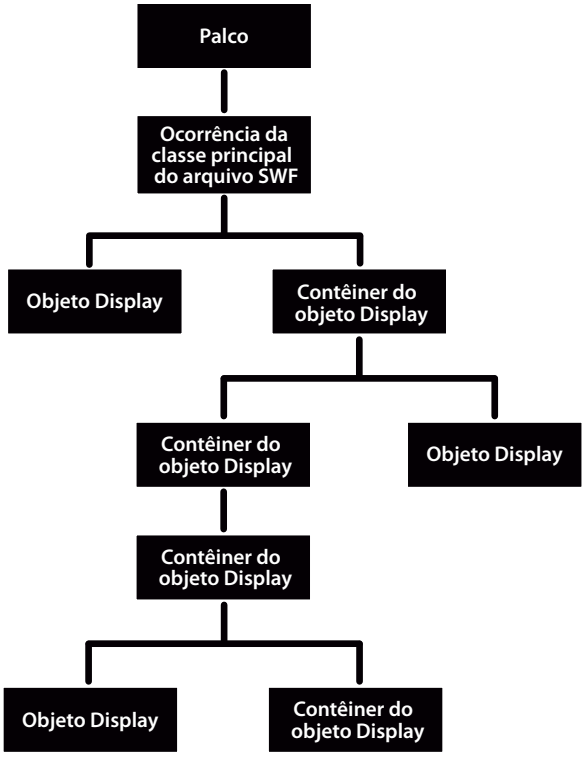
Os elementos visuais são programados no Adobe® ActionScript® 3.0 funcionando com objetos de exibição no palco correspondente. Por exemplo, você pode adicionar, mover, remover e solicitar objetos de exibição, aplicar filtros e máscaras, desenhar gráficos de vetores e bitmap, e realizar transformações em três dimensões usando a API de programação da exibição do ActionScript. As classes primárias utilizadas para programação de exibição são parte do pacote flash.display.

Nota: *O Adobe® AIR™ fornece o objeto HTMLLoader para renderizar e exibir o conteúdo em HTML. O HTMLLoader renderiza os elementos visuais do DOM HTML como um único objeto de exibição. Você não pode acessar os elementos individuais do DOM diretamente através da hierarquia da lista de exibição do ActionScript. Em vez disso, você acessa esses elementos DOM usando a API DOM separada fornecida pelo HTMLLoader.*

Noções básicas sobre a programação de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cada aplicativo desenvolvido no ActionScript 3.0 tem uma hierarquia de objetos exibidos, conhecida como *lista de exibição*, apresentada a seguir. A lista de exibição contém todos os elementos visíveis do aplicativo.



Conforme mostrado na ilustração, os elementos de exibição podem pertencer a um ou mais dos seguintes grupos:

- O Palco

  O Palco é o contêiner básico de objetos de exibição. Cada aplicativo tem um objeto Stage, que contém todos os objetos de exibição da tela. O Palco é o contêiner de nível superior e está no topo da hierarquia da lista de exibição:

  Cada arquivo SWF tem uma classe ActionScript associada, conhecida como *a classe principal do arquivo SWF*. Quando um arquivo SWF é aberto no Flash Player ou no Adobe AIR, o Flash Player ou o AIR chama a função do construtor dessa classe, e a ocorrência criada (que sempre é um tipo do objeto de exibição) é adicionada como filho do objeto Stage. A classe principal de um arquivo SWF sempre amplia a classe Sprite (para obter mais informações, consulte "Vantagens da abordagem da lista de exibição" na página 158).

  Você pode acessar o palco por meio da propriedade `stage` de qualquer ocorrência de DisplayObject. Para obter mais informações, consulte "Configuração de propriedades do palco" na página 167.

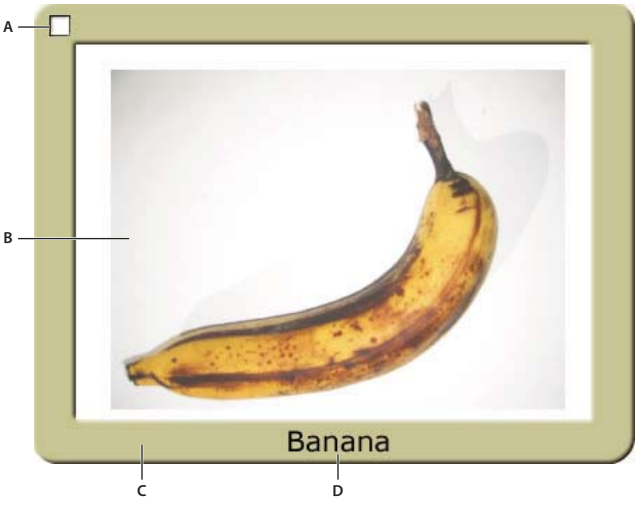
- Objetos de exibição

  No ActionScript 3.0, todos os elementos que aparecem na tela de um aplicativo são tipos de *objetos de exibição*. O pacote flash.display contém uma classe DisplayObject, que é uma classe-base estendida por uma série de outras classes. Essas classes diferentes representam tipos diferentes de objetos de exibição, como formas vetoriais, clipes de filme e campos de texto, entre outros. Para obter uma visão geral dessas classes, consulte "Vantagens da abordagem da lista de exibição" na página 158.

- Contêineres de objeto de exibição

  Os contêineres são tipos especiais de objeto de exibição que, além de ter sua própria representação visual, podem conter objetos filho que também são objetos de exibição.

  A classe DisplayObjectContainer é uma subclasse da classe DisplayObject. Um objeto DisplayObjectContainer pode conter vários objetos de exibição em sua *lista defilhos*. Por exemplo, a ilustração a seguir mostra um tipo de objeto DisplayObjectContainer conhecido como Sprite que contém vários objetos de exibição:



***A.** Um objeto SimpleButton. Esse tipo de objeto de exibição tem estados "up", "down" e "over" diferentes. **B.** Um objeto Bitmap. Nesse caso, o objeto Bitmap foi carregado a partir de um JPEG externo por meio de um objeto Loader. **C.** Um objeto Shape. O "quadro de imagem" contém um retângulo arredondado que é desenhado no ActionScript. Esse objeto Shape tem o filtro Sombra projetada aplicado. **D.** Um objeto TextField.*

No âmbito da discussão sobre objetos de exibição, os objetos DisplayObjectContainer também são conhecidos como *contêineres de objeto de exibição* ou simplesmente *contêineres*. Conforme mencionado anteriormente, o palco é um contêiner de objetos de exibição.

Embora todos os objetos de exibição visíveis sejam herdados da classe DisplayObject, o tipo de cada um é de uma subclasse específica da classe DisplayObject. Por exemplo, existe uma função de construtor para as classes Shape ou Video, mas não existe nenhum função assim para a classe DisplayObject.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você vai encontrar ao conhecer o gráfico de programação do ActionScript:

Alfa O valor de cor que representa a quantidade de transparência (melhor dizendo, a quantidade de opacidade) de uma cor. Por exemplo, uma cor com um canal alfa de 60% mostra apenas 60% de sua intensidade total e é 40% transparente.

Gráfico de bitmap Um elemento gráfico que é definido no computador como uma grade (linhas e colunas) de pixels coloridos. Em geral, os gráficos de bitmap incluem fotos digitais e imagens similares.

Modo de mesclagem Uma especificação de como o conteúdo de duas imagens sobrepostas deve interagir. Normalmente, uma imagem opaca em cima de outra imagem simplesmente bloqueia a imagem subjacente para que não fique visível. No entanto, modos de mesclagem diferentes fazem com que as cores das imagens se misturem de formas diferentes, de modo que o conteúdo resultante é uma combinação das duas imagens.

Lista de exibição A hierarquia de objetos de exibição que será renderizada como o conteúdo visível na tela pelo Flash Player e pelo AIR. O palco é a raiz da lista de exibição e todos os objetos de exibição anexados ao palco ou a um de seus filhos formam a lista de exibição (mesmo que o objeto não seja realmente renderizado, por exemplo, se estiver fora dos limites do palco).

Objeto de exibição Um objeto que representa algum tipo de conteúdo visual no Flash Player ou no AIR. Apenas objetos de exibição podem ser incluídos na lista de exibição e todas as classes desses objetos são subclasses de DisplayObject.

Contêiner do objeto de exibição Um tipo especial de objeto de exibição que pode conter objetos de exibição filhos além de (geralmente) ter sua própria representação visual.

Classe principal do arquivo SWF A classe que define o comportamento do objeto de exibição mais externo de um arquivo SWF e que, conceitualmente, é a classe do próprio arquivo SWF. Por exemplo, em um SWF criado com autoria em Flash, a classe principal é a classe document. Ela contém uma “linha do tempo principal” que contém todas as outras linhas do tempo; a classe principal do arquivo SWF é a classe da qual a linha do tempo principal é uma ocorrência.

Mascaramento Técnica para ocultar algumas partes de uma imagem (ou de exibir apenas algumas partes da imagem). As partes mascaradas da imagem ficam transparentes, de modo que o conteúdo subjacente é exibido. O termo está relacionado à fita isolante do pintor que é usada para impedir que algumas áreas sejam pintadas.

Palco O contêiner visual que é a base ou o plano de fundo de todo o conteúdo visual de um SWF.

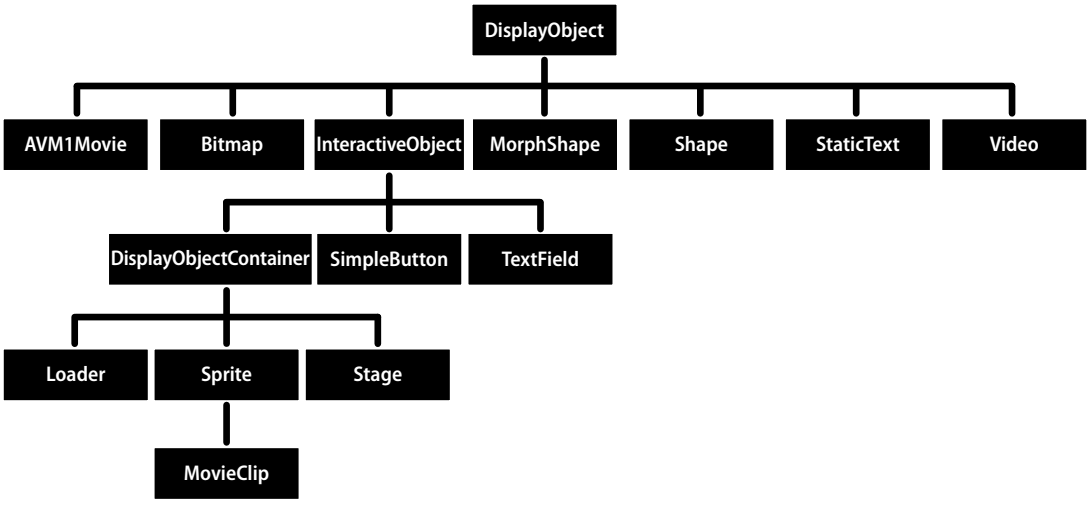
Transformação Ajuste de uma característica visual de um gráfico como a rotação do objeto, a alteração da escala, a inclinação ou distorção da forma ou a alteração da cor.

Gráfico vetorial Elemento gráfico que é definido no computador como linhas e formas desenhadas com características específicas (como espessura, comprimento, tamanho, ângulo e posição).

Principais classes de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O pacote flash.display do ActionScript 3.0 inclui classes para objetos visuais que podem aparecer no Flash Player ou no AIR. A ilustração a seguir mostra as relações de subclasse dessas classes principais de objeto de exibição.



A ilustração mostra a herança das classes de objeto de exibição. Algumas dessas classes, especificamente StaticText, TextField e Video, não estão no pacote flash.display, mas ainda são herdadas da classe DisplayObject.

Todas as classes derivadas de DisplayObject herdam seus métodos e propriedades. Para obter mais informações, consulte "Propriedades e métodos da classe DisplayObject" na página 161.

É possível percorrer os objetos das seguintes classes contidas no pacote flash.display:

- Bitmap - A classe Bitmap pode ser usada para definir objetos de bitmap, carregados a partir de arquivos externos ou renderizados pelo ActionScript. É possível carregar bitmaps a partir de arquivos externos por meio da classe Loader. É possível carregar arquivos GIF, JPG ou PNG. Além disso, você pode criar um objeto BitmapData com dados personalizados e criar um objeto Bitmap que usa esses dados. Você pode usar os métodos da classe BitmapData para alterar bitmaps, sejam eles carregados ou criados no ActionScript. Para obter mais informações, consulte "Carregamento de objetos de exibição" na página 202 e "Trabalho com bitmaps" na página 247.
- Loader - A classe Loader pode ser usada para carregar ativos externos (arquivos SWF ou elementos gráficos). Para obter mais informações, consulte "Carregamento dinâmico do conteúdo da exibição" na página 202.
- Shape - A classe Shape pode ser usada para criar gráficos vetoriais, como retângulos, linhas, círculos e assim por diante. Para obter mais informações, consulte "Uso da API de desenho" na página 226.
- SimpleButton - Um objeto SimpleButton é a representação do ActionScript do símbolo de um botão criado com a ferramenta de autoria do Flash. Uma ocorrência de SimpleButton tem quatro estados de botão: para cima, para baixo, por cima e teste de ocorrência (a área que responde aos eventos de mouse e de teclado).
- Sprite - Um objeto Sprite pode conter gráficos próprios e objetos de exibição filho. A classe Sprite estende a classe DisplayObjectContainer. Para obter mais informações, consulte "Trabalho com contêineres de objeto de exibição" na página 161 e "Uso da API de desenho" na página 226.

- MovieClip - Um objeto MovieClip é a forma atribuída pelo ActionScript a um símbolo de clipe de filme criado com a ferramenta de autoria do Flash. Na prática, o objeto MovieClip é similar a um objeto Sprite, mas também tem uma linha do tempo. Para obter mais informações, consulte "Trabalho com clipes de filme" na página 331.

As classes a seguir, que não estão no pacote flash.display, são subclasses de DisplayObject:

- A classe TextField, incluída no pacote flash.text, é um objeto de exibição para entrada e exibição de texto. Para obter mais informações, consulte "Princípios básicos do trabalho com texto" na página 381.
- A classe TextLine, incluída no pacote flash.text.engine, é o objeto de exibição usado para exibir linhas de texto compostas pelo Flash Text Engine e pelo Text Layout Framework. Para obter mais informações, consulte "Uso do Flash Text Engine" na página 408 e "Uso da Text Layout Framework" na página 438.
- A classe Video, incluída no pacote flash.media, é um objeto de exibição usado para exibir arquivos de vídeo. Para obter mais informações, consulte "Trabalho com vídeo" na página 487.

As seguintes classes do pacote flash.display estendem a classe DisplayObject, mas não é possível criar ocorrências delas. Em vez disso, elas servem como classes pai para outros objetos de exibição, combinando funcionalidades comuns em uma única classe.

- AVM1Movie - A classe AVM1Movie é usada para representar arquivos SWF carregados que são criados no ActionScript 1.0 e 2.0.
- DisplayObjectContainer - As classes Loader, Stage, Sprite e MovieClip estendem a classe DisplayObjectContainer. Para obter mais informações, consulte "Trabalho com contêineres de objeto de exibição" na página 161.
- InteractiveObject - InteractiveObject é a classe base de todos os objetos usados para interagir com o mouse e o teclado. Os objetos SimpleButton, TextField, Loader, Sprite, Stage e MovieClip são subclasses de InteractiveObject. Para obter mais informações sobre como criar a interação do mouse e do teclado, consulte "Princípios básicos da interação do usuário" na página 573.
- MorphShape - Esses objetos são criados quando você cria uma interpolação de forma na ferramenta de criação do Flash. Não é possível percorrê-los usando o ActionScript, mas é possível acessá-los a partir da lista de exibição.
- Stage - A classe Stage estende a classe DisplayObjectContainer. Existe uma ocorrência de Stage em cada aplicativo, que fica no topo da hierarquia da lista de exibição. Para acessar o palco, use a propriedade `stage` de qualquer ocorrência de DisplayObject. Para obter mais informações, consulte "Configuração de propriedades do palco" na página 167.

Além disso, a classe StaticText do pacote flash.text estende a classe DisplayObject, mas não é possível criar uma ocorrência dela no código. Os campos de texto estático só podem ser criados no Flash.

As classes a seguir não são objetos de exibição nem contêineres de objeto de exibição, e não constam da lista de exibição, apesar de exibirem elementos gráficos no palco. Essas classes desenham em um retângulo, chamado viewport, posicionado relativamente ao palco.

- StageVideo – A classe StageVideo exibe conteúdo de vídeo usando aceleração de hardware, quando possível. Essa classe está disponível no Flash Player 10.2. Para obter mais informações, consulte "Uso da classe StageVideo para apresentação acelerada por hardware" na página 528.
- StageWebView – A classe StageWebView exibe conteúdo HTML. Esta classe está disponível a partir do AIR 2.5. Para obter mais informações, consulte "Objetos StageWebView" na página 1063.

As seguintes classes fl.display proporcionam funcionalidade semelhante às das classes flash.display.Loader e LoaderInfo. Use essas classes em vez de suas contrapartidas flash.display se estiver desenvolvendo no ambiente Flash Professional (CS5.5 ou posterior). Nesse ambiente, essas classes ajudam a solucionar problemas que envolvam TLF com pré-carregamento de RSL. Para obter mais informações, consulte "Uso das classes ProLoader e ProLoaderInfo" na página 206.

- fl.display.ProLoader—Análoga a flash.display.Loader
- fl.display.ProLoaderInfo—Análoga a flash.display.LoaderInfo

Vantagens da abordagem da lista de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 3.0, existem classes separadas para diferentes tipos de objeto de exibição. No ActionScript 1.0 e 2.0, muitos objetos do mesmo tipo são incluídos em uma classe: a classe MovieClip.

Essa individualização de classes e a estrutura hierárquica das listas de exibição têm as seguintes vantagens:

- Renderização mais eficiente e menos uso da memória
- Melhor gerenciamento de profundidade
- Profundidade completa da lista de exibição
- Objetos de exibição fora da lista
- Facilidade de subclassificação dos objetos de exibição

Renderização mais eficiente e arquivos menores

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 1.0 e 2.0, você pode desenhar formas somente em um objeto MovieClip. No ActionScript 3.0, existem classes de objeto de exibição mais simples nas quais é possível desenhar formas. Como essas classes de objeto de exibição do ActionScript 3.0 não incluem um conjunto completo de métodos e propriedades como o objeto MovieClip, consomem menos recursos da memória e do processador.

Por exemplo, cada objeto MovieClip inclui propriedades para a linha do tempo do clipe de filme, enquanto o objeto Shape não faz isso. As propriedades de gerenciamento da linha do tempo podem usar muitos recursos da memória e do processador. No ActionScript 3.0, usar o objeto Shape melhora o desempenho. O objeto Shape tem uma sobrecarga menor do que o objeto MovieClip mais complexo. O Flash Player e o AIR não precisam gerenciar propriedades MovieClip não utilizadas, o que melhora a velocidade e reduz a memória usada pelo objeto.

Melhor gerenciamento de profundidade

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 1.0 e 2.0, a profundidade era gerenciada por um esquema linear e métodos como `getNextHighestDepth()`.

O ActionScript 3.0 inclui a classe DisplayObjectContainer, que tem métodos e propriedades mais práticos para gerenciar a profundidade dos objetos de exibição.

No ActionScript 3.0, ao mover um objeto de exibição para uma nova posição da lista de filhos de uma ocorrência de DisplayObjectContainer, os outros filhos do contêiner de objeto de exibição são reposicionados automaticamente e designados nas posições de índice secundárias apropriadas no contêiner.

Além disso, no ActionScript 3.0, sempre é possível detectar todos os objetos filho de qualquer contêiner de objeto de exibição. Cada ocorrência de DisplayObjectContainer tem uma propriedade `numChildren`, que lista o número de filhos no contêiner de objeto de exibição. Como a lista de filhos de um contêiner de objeto de exibição sempre é uma lista indexada, você pode examinar todos os objetos da lista desde a posição 0 até a última posição de índice (`numChildren - 1`). Isso não era possível com os métodos e as propriedades de um objeto MovieClip no ActionScript 1.0 e no 2.0.

No ActionScript 3.0, você pode percorrer com facilidade a lista de exibição na sequência; não existe nenhuma falha nos números de índice de uma lista de filhos de um contêiner de objeto de exibição. Percorrer a lista de exibição e gerenciar a profundidade dos objetos é muito mais fácil do que o que era permitido no ActionScript 1.0 e 2.0. No ActionScript 1.0 e 2.0, um clipe de filme podia ter objetos com lacunas intermitentes na ordem de profundidade, o que dificultava percorrer a lista de objetos. No ActionScript 3.0, cada lista de filhos de um contêiner de objeto de exibição é armazenada em cache internamente como uma matriz, resultando em consultas muito rápidas (por índice). Percorrer todos os filhos de um contêiner de objeto de exibição também é muito rápido.

No ActionScript 3.0, você também pode acessar os filhos de um contêiner de objeto de exibição usando o método `getChildByName()` da classe DisplayObjectContainer.

Profundidade completa da lista de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 1.0 e 2.0, não era possível acessar alguns objetos, como formas vetoriais, que eram desenhados na ferramenta de criação do Flash. No ActionScript 3.0, você pode acessar todos os objetos da lista de exibição - os criados com o ActionScript e todos os objetos criados na ferramenta de criação do Flash. Para obter detalhes, consulte "Como percorrer a lista de exibição" na página 165.

Objetos de exibição fora da lista

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 3.0, você pode criar objetos que não estão na lista de exibição visível. São os chamados objetos de exibição *fora da lista*. Um objeto de exibição é adicionado à lista visível somente quando você chama o método `addChild()` ou `addChildAt()` de uma ocorrência de DisplayObjectContainer que já foi adicionada à lista de exibição.

Você pode usar objetos de exibição fora da lista para montar objetos complexos, como os que têm vários contêineres com vários objetos de exibição. Mantendo os objetos de exibição fora da lista, você pode montar objetos complicados sem usar o tempo de processamento para renderizar esses objetos de exibição. Em seguida, você pode adicionar um objeto fora da lista à lista de exibição quando necessário. Além disso, é possível mover um filho de um contêiner de objeto de exibição na lista e fora dela e para qualquer posição desejada.

Facilidade de subclassificação dos objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 1.0 e 2.0, você tinha que adicionar com frequência novos objetos MovieClip a um arquivo SWF para criar formas básicas ou exibir bitmaps. No ActionScript 3.0, a classe DisplayObject inclui muitas subclasses internas, como Shape e Bitmap. Como as classes do ActionScript 3.0 são mais especializadas para tipos específicos de objetos, é mais fácil criar subclasses básicas das classes internas.

Por exemplo, para desenhar um círculo no ActionScript 2.0, era necessário criar uma classe CustomCircle que estende a classe MovieClip quando um objeto da classe personalizada é percorrido. No entanto, essa classe também incluía diversos métodos e propriedades da classe MovieClip (como `totalFrames`) não apropriados. No ActionScript 3.0, entretanto, é possível criar uma classe CustomCircle que estende o objeto Shape e, desse modo, não inclui métodos e propriedades não relacionados que estão contidos na classe MovieClip. O código a seguir mostra um exemplo de uma classe CustomCircle:

```
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);
    }
}
```

Trabalho com os objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Agora que você já conhece os conceitos básicos de palco, objetos de exibição, contêineres de objeto de exibição e lista de exibição, esta seção fornece informações mais específicas sobre como trabalhar com objetos de exibição no ActionScript 3.0.

Propriedades e métodos da classe DisplayObject

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Todos os objetos de exibição são subclasses de DisplayObject e, como tal, herda as propriedades e os métodos da classe DisplayObject. As propriedades herdadas são básicas e se aplicam a todos os objetos de exibição. Por exemplo, cada objeto de exibição tem uma propriedade `x` e uma propriedade `y` que especifica a posição do objeto no contêiner.

Não é possível criar uma ocorrência de DisplayObject usando o construtor da classe DisplayObject. Você deve criar outro tipo de objeto (que seja uma subclasse de DisplayObject), como Sprite, para percorrer um objeto com o operador `new`. Além disso, se desejar criar uma classe personalizada de objeto de exibição, você deve criar uma subclasse de uma das subclasses que têm uma função de construtor útil (como a classe Shape ou a classe Sprite). Para obter mais informações, consulte descrição de classeDisplayObject em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Adição de objetos de exibição à lista de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao ser percorrido, o objeto de exibição não aparece na tela (no palco) até que você adicione a ocorrência desse objeto a um contêiner que está na lista de exibição. Por exemplo, no código a seguir, o objeto TextField `myText` não ficaria visível se a última linha do código fosse omitida. Na última linha do código, a palavra-chave `this` deve fazer referência a um contêiner já adicionado à lista de exibição.

```
import flash.display.*;
import flash.text.TextField;
var myText:TextField = new TextField();
myText.text = "Buenos dias.";
this.addChild(myText);
```

Quando algum elemento visual é adicionado ao palco, esse elemento se transforma em *filho* do objeto Stage. O primeiro arquivo SWF carregado em um aplicativo (por exemplo, aquele incorporado em uma página HTML) é adicionado automaticamente como filho do objeto Stage. Pode ser um objeto de qualquer tipo que estende a classe Sprite.

Todos os objetos de exibição criados *sem* o ActionScript (por exemplo, por meio da inserção de uma tag MXML em um arquivo Flex MXML ou da colocação de um item no Palco no Flash Professional) são adicionados à lista de exibição. Embora esses objetos de exibição não sejam adicionados por meio do ActionScript, é possível acessá-los usando o ActionScript. Por exemplo, o código a seguir ajusta a largura de um objeto chamado `button1` que foi inserido na ferramenta de criação (não por meio do ActionScript):

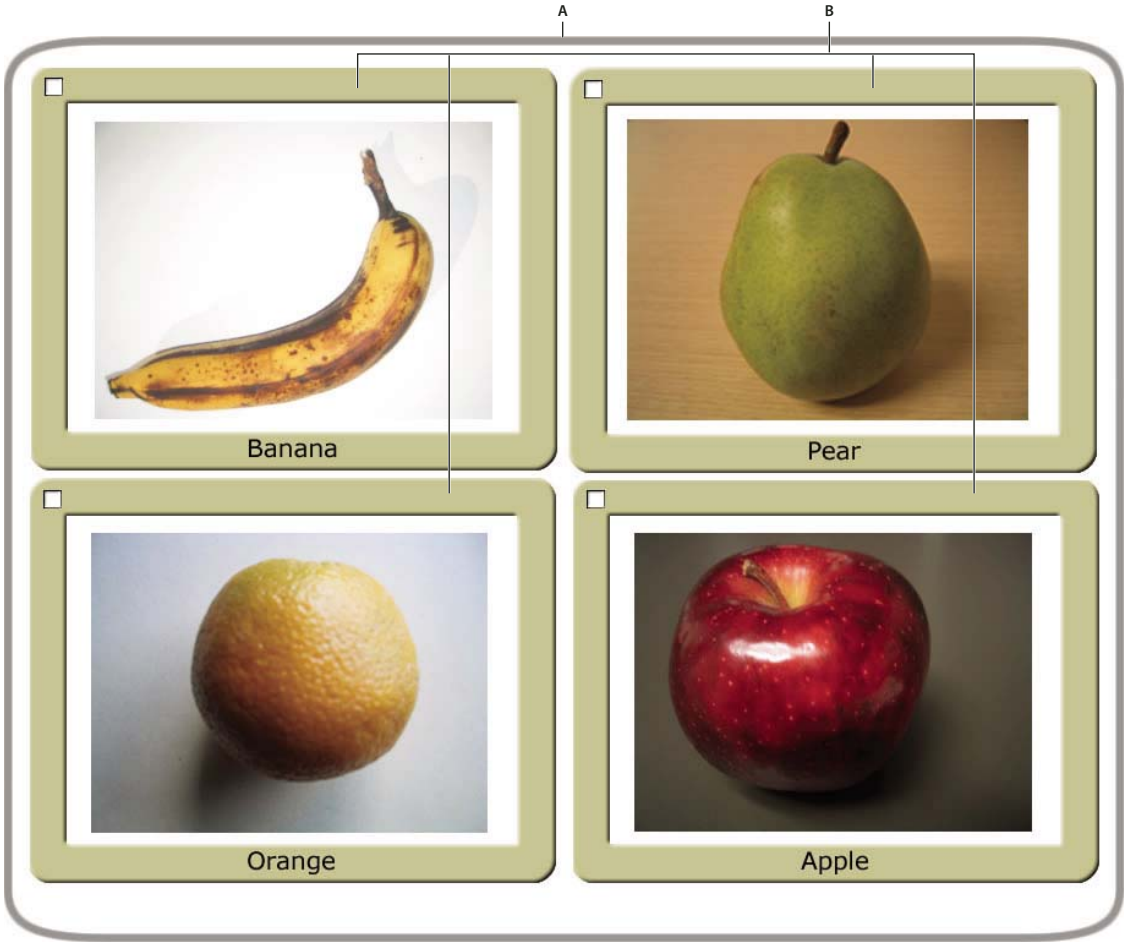
```
button1.width = 200;
```

Trabalho com contêineres de objeto de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se um objeto DisplayObjectContainer for excluído da lista de exibição, ou movido ou transformado de algum modo, cada objeto de exibição de DisplayObjectContainer também será excluído, movido ou transformado.

Um contêiner é um tipo de objeto de exibição propriamente dito e pode ser adicionado a outro contêiner. Por exemplo, a imagem a seguir mostra um contêiner de objeto de exibição, `pictureScreen`, que tem uma forma de contorno e outros quatro contêineres (do tipo PictureFrame):



A. *Uma forma que define a borda do contêiner de objeto de exibição pictureScreen* ***B.*** *Quatro contêineres de objeto de exibição que são filhos do objeto pictureScreen*

Para que um objeto apareça na lista de exibição, você deve adicioná-lo a um contêiner que esteja na lista. Para fazer isso, use o método `addChild()` ou o método `addChildAt()` do objeto de contêiner. Por exemplo, sem a linha final do código a seguir, o objeto `myTextField` não seria exibido:

```
var myTextField:TextField = new TextField();
myTextField.text = "hello";
this.root.addChild(myTextField);
```

Nesse exemplo de código, `this.root` aponta para o contêiner de objeto de exibição MovieClip que tem o código. No seu código real, você pode especificar um contêiner diferente.

Use o método `addChildAt()` para adicionar o filho a uma posição específica na lista de filhos do contêiner de objeto de exibição. Essas posições de índice baseadas em zero da lista de filhos estão relacionadas à disposição em camadas (ordem da frente para trás) dos objetos de exibição. Por exemplo, considere os três objetos de exibição a seguir. Cada objeto foi criado a partir de uma classe personalizada chamada Ball.

A disposição em camadas desses objetos de exibição no contêiner pode ser ajustada com o método `addChildAt()`. Por exemplo, considere o seguinte código:

```
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);
```

Depois que este código é executado, os objetos de exibição são posicionados do seguinte modo no `objeto` DisplayObjectContainer do contêiner. Observe a disposição em camadas dos objetos.

Para reposicionar um objeto na parte superior da lista de exibição, basta adicioná-lo novamente à lista. Por exemplo, depois do código anterior, para mover `ball_A` até a parte superior da pilha, use esta linha do código:

```
container.addChild(ball_A);
```

Este código remove `ball_A` por completo do seu local na lista de exibição do `contêiner` e o adicionam novamente à parte superior da lista, que equivale a movê-lo até a parte superior da pilha.

Você pode usar o método `getChildAt()` para verificar a ordem de camadas dos objetos de exibição. O método `getChildAt()` retorna objetos filho de um contêiner com base no número de índice fornecido. Por exemplo, o código a seguir revela nomes de objetos de exibição em posições diferentes da lista de filhos do objeto DisplayObjectContainer do `contêiner`:

```
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_C
trace(container.getChildAt(2).name); // ball_B
```

Se você remover um objeto de exibição da lista de filhos do contêiner pai, os elementos superiores da lista serão movidos para uma posição inferior no índice filho. Por exemplo, continuando com o código anterior, o código a seguir mostra como o objeto de exibição que estava na posição 2 no DisplayObjectContainer do `contêiner` será movido para a posição 1 se um objeto de exibição inferior da lista de filhos for removido:

```
container.removeChild(ball_C);
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_B
```

Os métodos `removeChild()` e `removeChildAt()` não excluem uma ocorrência de objeto de exibição por completo. Eles simplesmente a removem da lista de filhos do contêiner. A ocorrência ainda pode ser mencionada por outra variável. Use o operador `delete` para remover um objeto por completo.

Como um objeto de exibição tem apenas um contêiner pai, você pode adicionar uma ocorrência de um objeto de exibição a somente um contêiner. Por exemplo, o código a seguir mostra que o objeto de exibição `tf1` pode existir em apenas um contêiner (nesse caso, um Sprite, que estende a classe DisplayObjectContainer):

```
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
```

Se você adicionar um objeto de exibição que está contido em um contêiner a outro contêiner, esse objeto será removido da lista de filhos do primeiro contêiner.

Além dos métodos descritos acima, a classe DisplayObjectContainer define vários métodos para trabalhar com objetos de exibição filho, incluindo os seguintes:

- `contains()`: determina se um objeto de exibição é filho de DisplayObjectContainer.
- `getChildByName()`: recupera um objeto de exibição por nome.
- `getChildIndex()`: retorna a posição de índice de um objeto de exibição.
- `setChildIndex()`: altera a posição de um objeto de exibição filho.
- `removeChildren()`: remove vários objetos de visualização filho.
- `swapChildren()`: alterna a ordem de frente para trás de dois objetos de exibição.

- `swapChildrenAt()`: alterna a ordem de frente para trás de dois objetos de exibição, especificados pelos valores de índice.

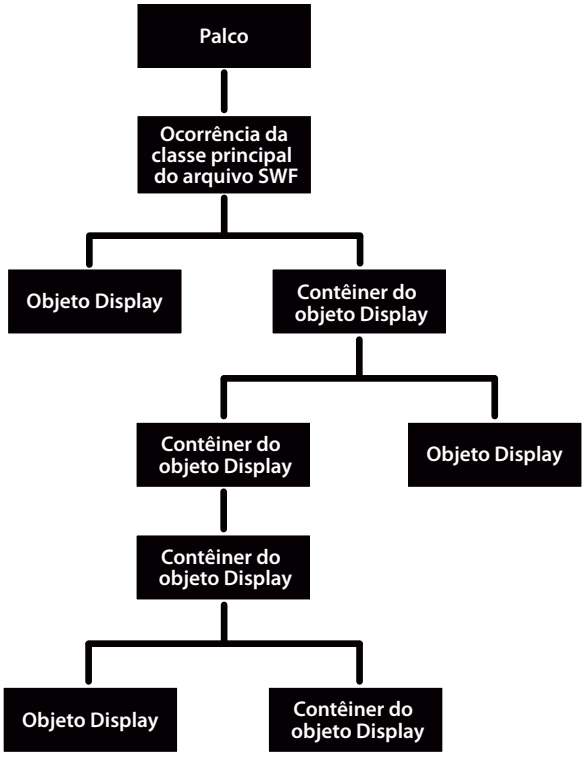
Para obter mais informações, consulte os respectivos itens na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Lembre-se de que um objeto de exibição que está fora da lista (não incluído em um contêiner de objeto de exibição que é filho do objeto Stage) é conhecido como objeto de exibição *fora da lista*.

Como percorrer a lista de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Como você observou, a lista de exibição é uma estrutura de árvore. Na parte superior da árvore está o palco, que pode conter vários objetos de exibição. Esses objetos que são contêineres propriamente ditos podem ter outros objetos de exibição ou contêineres.



A classe DisplayObjectContainer inclui propriedades e métodos para percorrer a lista de exibição por meio das listas de filhos dos contêineres de objeto de exibição. Por exemplo, considere o código a seguir, que adiciona dois objetos de exibição, `title` e `pict`, ao objeto `container` (que é um Sprite; a classe Sprite estende a classe DisplayObjectContainer):

```
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);
```

O método `getChildAt()` retorna o filho da lista de exibição em uma posição de índice específica:

```
trace(container.getChildAt(0) is TextField); // true
```

Você também pode acessar objetos filho por nome. Cada objeto de exibição tem uma propriedade de nome e, se você não designá-la, o Flash Player ou AIR atribuirá um valor padrão, como `"instance1"`. Por exemplo, o código a seguir mostra como usar o método `getChildByName()` para acessar um objeto de exibição filho com o nome `"banana loader"`:

```
trace(container.getChildByName("banana loader") is Loader); // true
```

O método `getChildByName()` piora mais o desempenho do que o método `getChildAt()`.

Como um contêiner de objeto de exibição pode ter outros contêineres como objetos filho em sua lista de exibição, é possível percorrer a lista inteira do aplicativo como uma árvore. Por exemplo, no trecho de código mostrado anteriormente, assim que a operação de carregamento do objeto `pict` Loader for concluída, o objeto `pict` terá um objeto de exibição filho, que é o bitmap carregado. Para acessar esse objeto de exibição de bitmap, você pode gravar `pict.getChildAt(0)`. Você também pode gravar `container.getChildAt(0).getChildAt(0)` (visto que `container.getChildAt(0) == pict`).

A função a seguir fornece uma saída `trace()` pretendida da lista de exibição a partir de um contêiner:

```
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

O Flex, caso você o utilize, define muitas classes de objeto de exibição e essas classes substituem os métodos de acesso à lista de exibição da classe DisplayObjectContainer. Por exemplo, a classe Container do pacote mx.core substitui o método `addChild()` e outros métodos da classe DisplayObjectContainer (estendida pela classe Container). No caso do método `addChild()`, a classe substitui o método de tal forma que não é possível adicionar todos os tipos de objeto de exibição a uma ocorrência de Container no Flex. O método substituído, nesse caso, exige que o objeto filho que está sendo adicionado seja do tipo mx.core.UIComponent.

Configuração de propriedades do palco

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Stage substitui a maioria das propriedades e dos métodos da classe DisplayObject. Se um desses métodos ou propriedades substituídos for chamado, o Flash Player e o AIR lançarão uma exceção. Por exemplo, o objeto Stage não tem as propriedades `x` ou `y`, pois sua posição é fixa como o principal contêiner do aplicativo. As propriedades `x` e `y` fazem referência à posição de um objeto de exibição com relação ao seu contêiner e, como o objeto Stage não está contido em nenhum outro contêiner, essas propriedades não são aplicáveis.

Nota: *Alguns métodos e propriedades da classe Stage estão disponíveis somente para objetos de exibição que estão na mesma caixa de proteção de segurança do primeiro arquivo SWF carregado. Para obter detalhes, consulte "segurança de Palco" na página 1103.*

Controle da taxa de quadros de reprodução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A propriedade `frameRate` da classe Stage é usada para definir a taxa de quadros de todos os arquivos SWF carregados no aplicativo. Para obter mais informações, consulte Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Controle do dimensionamento do palco

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando a parte da tela que representa o Flash Player ou o AIR é redimensionada, o tempo de execução ajusta automaticamente o conteúdo do palco para compensar. A propriedade `scaleMode` da classe Stage determina como o conteúdo do palco é ajustado. Essa propriedade pode ter quatro valores diferentes, definidos como constantes na classe flash.display.StageScaleMode:

- `StageScaleMode.EXACT_FIT` dimensiona o SWF de modo a preencher as novas dimensões do palco sem levar em conta a proporção original do conteúdo. Os fatores de escala podem não ser os mesmos para largura e altura, o que pode fazer com que o conteúdo pareça espremido ou alongado se a proporção do palco for alterada.
- `StageScaleMode.SHOW_ALL` dimensiona o SWF para fazê-lo caber inteiramente nas novas dimensões do palco sem alterar a proporção do conteúdo. Este modo de escala exibe todo o conteúdo, mas pode resultar em bordas "letterbox", como as barras pretas exibidas durante a exibição de um filme em widescreen em uma televisão padrão.
- `StageScaleMode.NO_BORDER` dimensiona o SWF de modo a preencher completamente as novas dimensões do palco sem alterar a proporção do conteúdo. Este modo de escala faz uso total da área de exibição do palco, mas pode resultar em cortes.
- `StageScaleMode.NO_SCALE` – não dimensiona o SWF. Se as novas dimensões do palco forem menores, o conteúdo será cortado; se forem maiores, o espaço adicionado ficará em branco.

  Somente no modo de escala `StageScaleMode.NO_SCALE`, as propriedades `stageWidth` e `stageHeight` da classe Stage podem ser usadas para determinar as dimensões de pixel reais do palco redimensionado. Nos outros modos de escala, as propriedades `stageWidth` e `stageHeight` sempre refletem a largura e a altura originais do SWF. Além disso, quando `scaleMode` é definido como `StageScaleMode.NO_SCALE` e o arquivo SWF é redimensionado, o evento `resize` da classe Stage é enviado e permite fazer os ajustes adequados.

Consequentemente, definir `scaleMode` como `StageScaleMode.NO_SCALE` permite que você tenha mais controle sobre o ajuste do conteúdo da tela para redimensionar a janela se desejar. Por exemplo, em um arquivo SWF que contém um vídeo e uma barra de controle, você talvez queira que a barra de controle permaneça do mesmo tamanho quando o palco for redimensionado e apenas o tamanho da janela do vídeo seja alterado para acomodar o novo tamanho do palco. Isso é demonstrado no exemplo a seguir:

```
// 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;
}
```

Definição do modo de dimensionamento do palco para as janelas do AIR

A propriedade de palco `scaleMode` determina como o palco é dimensionado e corta os objetos de exibição filho quando uma janela é redimensionada. Apenas o modo `noScale` deve ser usado no AIR. Nesse modo, o palco não é dimensionado. Ao contrário, o tamanho do palco muda diretamente os limites da janela. Os objetos podem ser cortados se a janela for redimensionada para um tamanho menor.

Os modos de dimensionamento de palco são criados para serem usados em ambientes como um navegador de Web, no qual você nem sempre pode controlar o tamanho ou a proporção de aspecto do palco. Os modos permitem que você faça a escolha mais adequada quando o palco não corresponde ao tamanho ideal nem à proporção de aspecto do seu aplicativo. No AIR, você sempre tem controle do palco de modo que, na maioria dos casos, a nova disposição do conteúdo ou o ajuste das dimensões na janela proporcionará melhores resultados do que ativar o dimensionamento do palco.

No navegador e na janela inicial do AIR, a relação entre o tamanho da janela e o fator de escala inicial é lida do arquivo SWF carregado. Entretanto, quando você cria um objeto NativeWindow, o AIR escolhe uma relação arbitrária entre o tamanho da janela e o fator de escala 72:1. Portanto, se a sua janela tem 72x72 pixels, um retângulo de 10x10 adicionado à janela é desenhado no tamanho correto de 10x10 pixels. Entretanto, se a janela tiver 144x144 pixels, um retângulo de 10x10 pixels será dimensionado para 20x20 pixels. Se você insistir em usar um `scaleMode` que não seja `noScale` para o palco de uma janela, poderá compensar definindo o fator de escala de qualquer objeto de exibição da janela na proporção de 72 pixels para a largura e a altura atuais do palco. Por exemplo, o código a seguir calcula o fator de escala necessário para um objeto de exibição denominado `client`:

```
if(newWindow.stage.scaleMode != StageScaleMode.NO_SCALE){
client.scaleX = 72/newWindow.stage.stageWidth;
client.scaleY = 72/newWindow.stage.stageHeight;
}
```

Nota: *As janelas Flex e HTML definem automaticamente o palco `scaleMode` como `noScale`. Alterar o `scaleMode` perturba os mecanismos automáticos de layout usados nesses tipos de janelas.*

Trabalho com o modo de tela cheia

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O modo de tela cheia permite que você configure o palco de um filme para preencher todo o monitor do visualizador, sem nenhuma borda ou menu de contêiner. A propriedade `displayState` da classe Stage é usada para ativar e desativar o modo de tela cheia para um SWF. A propriedade `displayState` pode ser configurada como um dos valores definidos pelas constantes da classe flash.display.StageDisplayState. Para ativar o modo de tela cheia, defina a propriedade `displayState` como `StageDisplayState.FULL_SCREEN`:

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

Para ativar o modo interativo de tela inteira (novidade no Flash Player 11.3), defina a propriedade `displayState` como `StageDisplayState.FULL_SCREEN_INTERACTIVE`:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

No Flash Player, o modo de tela cheia só pode ser iniciado por meio do ActionScript em resposta a um clique do mouse (incluindo o clique com o botão direito) ou pressionamento de tecla. O conteúdo do AIR em execução na caixa de proteção de segurança do aplicativo não requer a ativação do modo de tela cheia em resposta a um gesto do usuário.

Para ativar o modo de tela cheia, defina a propriedade `displayState` como `StageDisplayState.NORMAL`.

```
stage.displayState = StageDisplayState.NORMAL;
```

Além disso, o usuário pode optar por sair do modo de tela cheia alternando o foco para outra janela ou usando uma de várias combinações de tecla: a tecla Esc (todas as plataformas), Control-W (Windows), Command-W (Mac) ou Alt-F4 (Windows).

Ativação do modo de tela cheia no Flash Player

Para ativar o modo de tela cheia para um arquivo SWF incorporado a uma página HTML, o código HTML a ser incorporado ao Flash Player deve incluir uma tag `param` e o atributo `embed` com o nome `allowFullScreen` e o valor `true`, do seguinte modo:

```
<object>
    ...
    <param name="allowFullScreen" value="true" />
    <embed ... allowFullScreen="true" />
</object>
```

Na ferramenta de autoria do Flash, selecione Arquivo -> Configurações de publicação e, na aba HTML da caixa de diálogo Configurações de publicação, selecione o modelo Somente Flash - Permitir tela cheia.

No Flex, verifique se o modelo HTML inclui as tags `<object>` e `<embed>` que dão suporte a tela cheia.

Se estiver usando JavaScript em uma página da Web para gerar as tags incorporadas no SWF, altere o JavaScript para adicionar a tag `allowFullScreen param` e o atributo. Por exemplo, se a página HTML usa a função `AC_FL_RunContent()` (que é usada pelas páginas HTML geradas pelo Flash Professional e pelo Flash Builder), adicione o parâmetro `allowFullScreen` a essa chamada de função do seguinte modo:

```
AC_FL_RunContent(
    ...
    'allowFullScreen','true',
    ...
    ); //end AC code
```

Isso não se aplica aos arquivos SWF em execução no Flash Player autônomo.

Nota: *Se você definir o Modo de janela (wmode no HTML) como Opaco sem janela (opaco) ou Transparente sem janela (transparente), a janela da tela cheia será sempre opaca*

O uso do modo de tela cheia com o Flash Player em um navegador também apresenta restrições de segurança. Essas restrições estão descritas em "Segurança" na página 1079.

Ativação do modo interativo de tela inteira no Flash Player 11.3 e superior

O Flash Player 11.3 e superior suporta o modo interativo de tela inteira, o que permite o suporte integral para todas as teclas do teclado (exceto **Esc**, que sai do modo interativo de tela inteira). Moto interativo de tela cheia útil para jogos (por exemplo, para permitir o bate papo em um jogo multijogador ou controles de teclado WASD em um jogo de tiro em primeira pessoa).

Para ativar o modo interativo de tela inteira para um arquivo SWF incorporado em uma página HTML, o código HTML para incorporar o Flash Player deve incluir uma marca `param` e o atributo `embed` com o nome `allowFullScreenInteractive` e o valor `true` da seguinte forma:

```
<object>
    ...
    <param name="allowFullScreenInteractive" value="true" />
    <embed ... allowFullScreenInteractive="true" />
</object>
```

Na ferramenta de autoria do Flash, selecione Arquivo -> Configurações de publicação e, na aba HTML da caixa de diálogo Configurações de publicação, selecione o modelo Somente Flash - Permitir tela cheia.

No Flash Builder e Flex, verifique se os modelos HTML incluem as marcas de formatação `<object>` e `<embed>` que suportam o modo interativo de tela inteira.

Se estiver usando o JavaScript em uma página da Web para gerar as marcas de formatação integradas SWF, você deve alterar o JavaScript para adicionar a marca de formatação e o atributo `allowFullScreenInteractive param`. Por exemplo, se a página HTML usa a função `AC_FL_RunContent()` (que é usada em páginas HTML geradas pelo Flash Professional e pelo Flash Builder), você deve adicionar o parâmetro `allowFullScreenInteractive` para a função de chamada da seguinte forma:

```
AC_FL_RunContent(
    ...
    'allowFullScreenInteractive','true',
    ...
    ); //end AC code
```

Isso não se aplica aos arquivos SWF em execução no Flash Player autônomo.

Tamanho do palco e dimensionamento em tela cheia

As propriedades `Stage.fullScreenHeight` e `Stage.fullScreenWidth` retornam a altura e a largura do monitor utilizado quando se passa para o tamanho de tela cheia, caso a entrada nesse estado ocorra imediatamente. Esses valores poderão estar incorretos se o usuário tiver a oportunidade de mudar o navegador de um monitor para outro depois que você recuperá-los, mas antes de entrar no modo de tela cheia. Se você recuperar esses valores no mesmo manipulador de eventos em que definiu a propriedade `Stage.displayState` como `StageDisplayState.FULL_SCREEN`, eles estarão corretos. No caso de usuários que têm diversos monitores, o conteúdo SWF se expande para ocupar apenas um monitor. O Flash Player e o AIR usam uma métrica para determinar qual monitor contém a maior parte do SWF e usam esse monitor para o modo de tela cheia. As propriedades fullScreenHeight e fullScreenWidth refletem apenas o tamanho do monitor usado no modo de tela cheia. Para obter mais informações, consulte `Stage.fullScreenHeight` e `Stage.fullScreenWidth` em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

O comportamento de dimensionamento do palco para o modo de tela cheia é igual ao de um modo normal; o dimensionamento é controlado pela propriedade `scaleMode` da classe Stage. Se a propriedade `scaleMode` estiver definida como `StageScaleMode.NO_SCALE`, as propriedades `stageWidth` e `stageHeight` da classe Stage serão alteradas para refletir o tamanho da área da tela ocupado pelo SWF (a tela inteira, nesse caso); se for visualizado no navegador, o parâmetro HTML controlará a configuração.

Você pode usar o evento `fullScreen` da classe Stage para detectar e responder quando o modo de tela cheia está ativado ou desativado. Por exemplo, você talvez queira reposicionar, adicionar ou remover itens da tela ao acessar ou sair do modo de tela cheia, como mostra este exemplo:

```
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);
```

Como mostra este código, o objeto do evento `fullScreen` é uma ocorrência da classe flash.events.FullScreenEvent, que inclui uma propriedade `fullScreen` que indica se o modo de tela cheia está ativado (`true`) ou não (`false`).

Suporte para teclado no modo de tela cheia

Quando o Flash Player é executado em um navegador, todo ActionScript relacionado ao teclado, como eventos de teclado e entrada de texto nas ocorrências de TextField, é desativado no modo de tela cheia. As exceções (as teclas que ficam ativadas) são estas:

- Teclas selecionadas que não são impressas, especificamente as teclas de seta, a barra de espaço e a tecla Tab
- Os atalhos de teclado que encerram o modo de tela cheia: Esc (Windows e Mac), Control-W (Windows), Command-W (Mac) e Alt-F4

Essas restrições não se aplicam a conteúdo SWF em execução no Flash Player independente ou no AIR. O AIR dá suporte a um modo de tela cheia interativa que permite entrada do teclado.

Suporte para mouse no modo de tela cheia

Por padrão, os eventos do mouse no modo de tela cheia funcionam da mesma forma dos eventos fora do modo de tela cheia. No entanto, no modo de tela cheia, você pode, como opção, configurar a propriedade `Stage.mouseLock` para ativar o travamento do mouse. O travamento do mouse desativa o cursor e permite o movimento sem restrições do mouse.

Nota: *Você somente pode ativar o travamento do mouse no modo de tela cheia para aplicativos de área de trabalho. Se você configurar em aplicativos que não estejam no modo de tela cheia ou em aplicativos em dispositivos móveis, criará uma exceção.*

O bloqueio do mouse é automaticamente desativado e o cursor torna-se visível novamente quando:

- O usuário sai do modo de tela inteira usando a tecla Esc (todas as plataformas), Ctrl+W (Windows), Comando+W (Mac) ou Alt+F4 (Windows).
- A janela do aplicativo fica desfocada.
- Qualquer UI de configuração é visível, incluindo todas as caixas de diálogo de privacidade.
- Uma caixa de diálogo nativa é mostrada, por exemplo, uma caixa de diálogo de carregamento de arquivo.

Eventos associados a movimentos do mouse, como o evento `mouseMove`, use a classe MouseEvent para representar o objeto do evento. Quando o travamento do mouse estiver desativado, use as propriedades `MouseEvent.localX` e `MouseEvent.localY` para determinar a localização do mouse. Quando o travamento do mouse estiver ativado, use as propriedades `MouseEvent.movementX` e `MouseEvent.movementY` para determinar a localização do mouse. As propriedades `movementX` e `movementY` contêm alterações na posição do mouse desde o último evento, em vez das coordenadas absolutas da localização do mouse.

Dimensionamento em hardware no modo de tela cheia

Você pode usar a propriedade `fullScreenSourceRect` da classe Stage para configurar o Flash Player ou o AIR para dimensionar uma região específica do palco no modo de tela cheia. O Flash Player e o AIR fazem dimensionamento em hardware, se possível, usando a placa gráfica e de vídeo do computador de um usuário, e geralmente exibe conteúdo mais rapidamente do que no dimensionamento em software.

Para tirar vantagem do dimensionamento em hardware, defina o palco inteiro ou parte dele para o modo de tela cheia. O código ActionScript 3.0 a seguir define o palco inteiro para o modo de tela cheia:

```
import flash.geom.*;
{
    stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

Quando essa propriedade é definida como um retângulo válido e a propriedade `displayState` é definida como o modo de tela cheia o Flash Player e o AIR dimensionam a área especificada. O tamanho real do Palco em pixels no ActionScript não é alterado. O Flash Player e o AIR forçam um limite mínimo para o tamanho do retângulo de forma a acomodar a mensagem padrão "Pressione Esc para sair do modo de tela cheia". Em geral, esse limite está em torno de 260 por 30 pixels, mas pode variar de acordo com a plataforma e a versão do Flash Player.

A propriedade `fullScreenSourceRect` só pode ser definida quando o Flash Player ou AIR não está no modo de tela cheia. Para usar esta propriedade corretamente, defina-a primeiro e, em seguida, defina a propriedade `displayState` para o modo de tela cheia.

Para ativar o dimensionamento, defina a propriedade `fullScreenSourceRect` como um objeto de retângulo.

```
stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
```

Para desativar o dimensionamento, defina a propriedade `fullScreenSourceRect` como `null`.

```
stage.fullScreenSourceRect = null;
```

Para aproveitar todos os recursos de aceleração de hardware com o Flash Player, ative-a na caixa de diálogo Configurações do Flash Player. Para carregar a caixa de diálogo, clique com o botão direito do mouse (Windows) ou clique mantendo a tecla Control pressionada (Mac) dentro do conteúdo Flash Player exibido no navegador. Selecione a aba Exibir, que é a primeira, e clique na caixa de seleção: Habilitar aceleração de hardware.

Modos de janela direto e de composição GPU

O Flash Player 10 introduz dois modos de janela (direto e composição GPU) que você pode ativar através das configurações de publicação da ferramenta de autoria do Flash. Esses modos não são suportados no AIR. Para aproveitar as vantagens desses modos, você deve ativar a aceleração de hardware para o Flash Player.

O modo direto usa o caminho mais rápido e direto para enviar gráficos para a tela, o que é vantajoso para a reprodução de vídeos.

A Composição GPU usa a unidade de processamento gráfico da placa de vídeo para acelerar a composição. A composição de vídeo é o processo de dispor várias imagens em camadas para criar uma única imagem de vídeo. Quando a composição é acelerada com a GPU, ela pode melhorar o desempenho de conversão YUV, correção de cores, rotação ou dimensionamento e mesclagem. Conversão YUV refere-se à conversão de cores de sinais analógicos compostos, que são usados para transmissão, no modelo de cores RGB (vermelho, verde, azul) usado por monitores e câmeras de vídeo. O uso da GPU para acelerar a composição reduz as demandas de memória e de computação que, de outro modo, seriam atribuídas à CPU. Ele também resulta em uma reprodução mais contínua para vídeo com definição padrão.

Seja cauteloso na implementação desses modos de janela. O uso da composição GPU pode ser dispendioso para a memória e os recursos da CPU. Se não for possível executar algumas operações (como modos de mesclagem, filtragem, corte ou mascaramento) na GPU, elas serão feitas pelo software. A Adobe recomenda que você se restrinja a um arquivo SWF por página HTML quando usar esses modos e que não ative esses modos para banners. O recurso Testar filme do Flash não utiliza aceleração de hardware, mas você pode usá-la através da opção Publicar visualização.

É inútil configurar uma taxa de quadros superior a 60 no arquivo SWF, que é a taxa máxima de atualização da tela. Configurar a taxa de quadros com um valor entre 50 e 55 viabiliza quadros descartados, que podem ocorrer por vários motivos de tempos em tempos.

O uso do modo direto exige o Microsoft DirectX 9 com 128 MB de VRAM no Windows e OpenGL para Apple Macintosh, Mac OS X v10.2 ou superior. A composição GPU exige o Microsoft DirectX 9 e suporte para Pixel Shader 2.0 no Windows com 128 MB de VRAM. No Mac OS X e no Linux, a composição GPU exige o OpenGL 1.5 e várias extensões OpenGL (objeto framebuffer, multitextura, objetos Shader, linguagem de sombreamento, sombreador de fragmentos).

É possível ativar os modos de aceleração `direto` e `gpu` por arquivo SWF na caixa de diálogo Configurações de publicação do Flash usando o menu Aceleração de hardware da aba Flash. Se você clicar em Nenhum, o modo de janela voltará para `padrão`, `transparente` ou `opaco`, conforme especificado pela configuração Modo de janela na aba HTML.

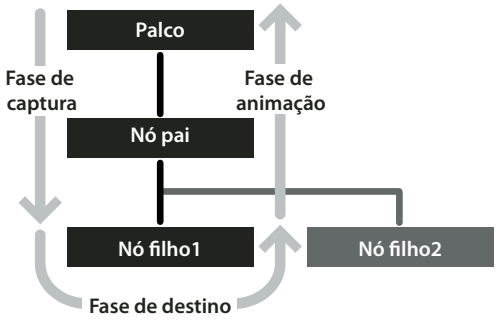
Manipulação de eventos de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe DisplayObject é herdada da classe EventDispatcher. Desse modo, todos os objetos de exibição podem participar plenamente no modelo de evento (descrito em "Manipulação de eventos" na página 126). Todos os objetos de exibição podem usar seu método `addEventListener()` (herdado da classe EventDispatcher) para ouvir um evento específico, mas somente se o objeto ouvinte fizer parte do fluxo desse evento.

Quando o Flash Player ou o AIR envia um objeto de evento, esse objeto faz uma viagem de ida e volta do palco até o objeto de exibição onde ocorreu o evento. Por exemplo, se o usuário clicar em um objeto de exibição chamado `child1`, o Flash Player enviará um objeto de evento do palco, por meio da hierarquia da lista de exibição, até o objeto de exibição `child1`.

O fluxo do evento é conceitualmente dividido em três fases, como mostra este diagrama:



Para obter mais informações, consulte "Manipulação de eventos" na página 126.

Uma questão importante que deve ser considerada ao trabalhar com eventos de objeto de exibição é o efeito que os ouvintes de evento podem ter quando os objetos de exibição removidos automaticamente da memória (lixo coletado) forem removidos da lista de exibição. Se um objeto de exibição tiver objetos inscritos como ouvintes de eventos, esse objeto não será removido da memória mesmo quando for removido da lista de exibição, pois ainda terá referências a esses objetos de ouvinte. Para obter mais informações, consulte "Gerenciamento de ouvintes de evento" na página 140.

Escolha de uma subclasse de DisplayObject

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Com tantas opções disponíveis, ao trabalhar com objetos de exibição, uma das decisões importantes é definir qual objeto será usado para cada finalidade. Veja algumas diretrizes que podem ajudá-lo a tomar essa decisão. Essas mesmas sugestões podem ser usadas quando você precisa de uma ocorrência de uma classe ou está escolhendo uma base para criar uma classe:

- Se você não precisar de um objeto que possa ser contêiner de outros objetos de exibição (isto é, precisar de apenas um que sirva como um elemento de tela autônomo), escolha uma dessas subclasses de DisplayObject ou InteractiveObject, dependendo do uso pretendido:
  - Bitmap para exibir uma imagem de bitmap.

  - TextField para adicionar texto.
  - Video para exibir vídeo.
  - Shape para uma “tela de desenho” para desenhar o conteúdo na tela. Especificamente, se desejar criar uma ocorrência para desenhar formas na tela que não será contêiner de outros objetos de exibição, use Shape em vez de Sprite ou MovieClip para melhorar o desempenho significativamente.
  - MorphShape, StaticText ou SimpleButton para itens criados com a ferramenta de autoria do Flash. Não é possível criar ocorrências dessas classes de modo programático, mas você pode criar variáveis com esses tipos de dados para fazer referência aos itens criados com a ferramenta da autoria do Flash.
- Se precisar de uma variável para fazer referência ao palco principal, use a classe Stage como tipo de dados.
- Se precisar de um contêiner para carregar um arquivo SWF ou de imagem externo, use uma ocorrência de Loader. O conteúdo carregado será adicionado à lista de exibição como filho da ocorrência de Loader. O tipo de dados depende da natureza do conteúdo carregado, do seguinte modo:
  - Uma imagem carregada será uma ocorrência do Bitmap.
  - Um arquivo SWF carregado gravado no ActionScript 3.0 será uma ocorrência de Sprite ou MovieClip (ou uma ocorrência de uma subclasse dessas classes, conforme especificado pelo criador do conteúdo).
  - Um arquivo SWF carregado gravado no ActionScript 1.0 ou no ActionScript 2.0 será uma ocorrência de AVM1Movie.
- Se precisar de um objeto para servir como contêiner de outros objetos de exibição (esteja você desenhando ou não no objeto de exibição com o ActionScript), escolha uma das subclasses de DisplayObjectContainer:
  - Sprite se o objeto for criado apenas com o ActionScript ou como a classe base de um objeto de exibição personalizado que será criado e manipulado somente com o ActionScript.
  - MovieClip se estiver criando uma variável para fazer referência a um símbolo de clipe de filme criado na ferramenta de criação do Flash.
- Se estiver criando uma classe que será associada a um símbolo de clipe de filme na biblioteca do Flash, escolha uma destas subclasses de DisplayObjectContainer como sua classe base:
  - MovieClip se o símbolo de clipe de filme associado tiver conteúdo em mais de um quadro
  - Sprite se o símbolo de clipe de filme associado tiver conteúdo somente no primeiro quadro

Manipulação de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Independentemente do objeto de exibição utilizado, existem diversas manipulações comuns a todos os objetos de exibição que servem como elementos exibidos na tela. Por exemplo, todos podem ser posicionados na tela, movidos para frente ou para trás na ordem de empilhamento dos objetos de exibição, dimensionados, girados e assim por diante. Como todos os objetos de exibição herdam essa funcionalidade da classe base comum (DisplayObject), tal funcionalidade apresenta o mesmo comportamento demonstrado na manipulação de uma ocorrência de TextField, de Video, de Shape ou de qualquer outro objeto de exibição. As seções a seguir descrevem em detalhes diversas manipulações comuns de objeto de exibição.

Alteração da posição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A manipulação mais básica de qualquer objeto de exibição é seu posicionamento na tela. Para definir a posição de um objeto de exibição, altere as propriedades `x` e `y` do objeto.

```
myShape.x = 17;
myShape.y = 212;
```

O sistema de posicionamento de objetos de exibição trata o palco como um sistema de coordenadas cartesianas (o sistema de grade comum com um eixo x horizontal e um eixo y vertical). A origem do sistema de coordenadas (a coordenada 0,0 onde os eixos x e y se encontram) está no canto superior esquerdo do palco. A partir desse ponto, os valores de x são positivos para a direita e negativos para a esquerda, enquanto (diferente dos sistemas gráficos típicos) os valores de y são positivos para baixo e negativos para cima. Por exemplo, as linhas anteriores do código movem o objeto `myShape` até a coordenada 17 do eixo x (17 pixel à direita da origem) e a coordenada 212 do eixo y (212 pixels abaixo da origem).

Por padrão, quando um objeto de exibição é criado com o ActionScript, as propriedades `x` e `y` são definidas como 0, colocando o objeto no canto superior esquerdo do conteúdo pai.

Alteração da posição em relação ao palco

É importante lembrar que as propriedades `x` e `y` sempre fazem referência à posição do objeto de exibição em relação à coordenada 0,0 dos eixos do objeto de exibição pai. Assim, para uma ocorrência de Shape (como um círculo) contida em uma ocorrência de Sprite, se você definir as propriedades `x` e `y` do objeto Shape como 0, o círculo será colocado no canto superior esquerdo de Sprite, que não é necessariamente o canto superior esquerdo do palco. Para posicionar um objeto em relação às coordenadas globais do palco, você pode usar o método `globalToLocal()` de qualquer objeto de exibição para converter coordenadas globais (palco) em locais (contêiner do objeto de exibição), do seguinte modo:

```
// 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;
```

Do mesmo modo, você pode usar o método `localToGlobal()` da classe DisplayObject para converter coordenadas locais em coordenadas do palco.

Movimento de objetos de exibição com o mouse

Você pode permitir que um usuário movimente objetos de exibição com o mouse, utilizando duas técnicas diferentes no ActionScript. Em ambos os casos, dois eventos de mouse são usados: quando o botão do mouse é pressionado, o objeto é acionado para seguir o cursor do mouse e, quando é solto, o objeto deve parar de segui-lo.

Nota: *O Flash Player 11.3 e superior, AIR 3.3 e superior: você também pode usar o evento MouseEvent.RELEASE_OUTSIDE para cobrir o caso de um usuário liberando o botão fora dos limites da entidade gráfica continente.*

A primeira técnica, usando o método `startDrag()`, é mais simples, porém mais limitada. Quando o botão do mouse é pressionado, o método `startDrag()` do objeto de exibição a ser arrastado é chamado. Quando o botão do mouse é solto, o método `stopDrag()` é chamado. A classe Sprite define essas duas funções. Portanto, o objeto movimentado deve ser um Sprite ou uma de suas subclasses.

```
// 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);
```

Essa técnica tem uma limitação bem significativa: somente um item por vez pode ser arrastado com `startDrag()`. Se um objeto de exibição estiver sendo arrastado e o método `startDrag()` for chamado em outro objeto de exibição, o primeiro objeto deixará de seguir o mouse imediatamente. Por exemplo, se a função `startDragging()` for alterada conforme mostrado aqui, somente o objeto `circle` será arrastado, apesar da chamada do método `square.startDrag()`:

```
function startDragging(event:MouseEvent):void
{
    square.startDrag();
    circle.startDrag();
}
```

Como apenas um objeto pode ser arrastado por vez com `startDrag()`, o método `stopDrag()` pode ser chamado em qualquer objeto de exibição e para sempre que o objeto está sendo arrastado.

Se precisar arrastar mais de um objeto de exibição ou para evitar a possibilidade de conflitos quando houver mais de um objeto que provavelmente use `startDrag()`, é melhor usar a técnica de acompanhamento do mouse para criar o efeito de desenho. Com essa técnica, quando o botão do mouse é pressionado, uma função é inscrita como ouvinte no evento `mouseMove` do palco. Essa função, que é chamada sempre que o mouse se move, faz com que o objeto arrastado passe para a coordenada x, y do mouse. Assim que o botão do mouse é solto, a função deixa de ser ouvinte, ou seja, não será mais chamada quando o mouse se movimentar, e o objeto para de acompanhar o cursor do mouse. Veja um código que demonstra essa técnica:

```
// 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);
```

Além de fazer com que um objeto de exibição acompanhe o cursor do mouse, muitas vezes é desejável mover o objeto arrastado até a frente da exibição para que apareça flutuando acima de todos os outros objetos. Por exemplo, suponha que você tenha dois objetos, um círculo e um quadrado, que possam ser movimentados com o mouse. Se o círculo ficar abaixo do quadrado na lista de exibição e você clicar e arrastar o círculo para que o cursor fique por cima do quadrado, irá parecer que o círculo desliza ao lado do quadrado, quebrando a ilusão de arrastar e soltar. Em vez disso, você pode definir que, quando for clicado, o círculo deve se mover para cima da lista de exibição e, assim, sempre aparecer em cima de qualquer outro conteúdo.

O código a seguir (adaptado do exemplo anterior) permite que dois objetos de exibição, um círculo e um quadrado, sejam movimentados com o mouse. Sempre que o botão do mouse é pressionado em um deles, esse item é movido para a parte superior da lista de exibição do palco para que o item arrastado sempre apareça por cima. (O código novo ou alterado com relação à listagem anterior é exibido em negrito.)

```
// 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);
```

Para estender esse efeito ainda mais, como para um jogo onde pinos ou cartas são movidos entre pilhas, você pode adicionar o objeto arrastado à lista de exibição do palco quando for "tirado" e adicioná-lo a outra lista de exibição (como a "pilha" onde é solto) quando o botão do mouse for liberado.

Finalmente, para aprimorar o efeito, você poderia aplicar um filtro de sombra projetada no objeto de exibição quando for clicado (quando começar a ser arrastado) e remover a sombra projetada quando o objeto for solto. Para obter detalhes sobre como usar o filtro de sombra projetada e outros filtros de objeto de exibição no ActionScript, consulte "Filtro de objetos de exibição" na página 274.

Visão panorâmica e rolagem de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se houver um objeto de exibição muito grande para a área na qual deseja exibi-lo, você pode usar a propriedade `scrollRect` para definir a área visível do objeto de exibição. Além disso, alterando a propriedade `scrollRect` em resposta à entrada do usuário, você pode obter uma visão panorâmica do conteúdo à esquerda e à direita e percorrer para cima e para baixo.

A propriedade `scrollRect` é uma ocorrência da classe Rectangle, que combina os valores necessários para definir uma área retangular como um único objeto. Para definir inicialmente a área visível do objeto de exibição, crie uma nova ocorrência de Rectangle e a atribua à propriedade `scrollRect` do objeto de exibição. Posteriormente, para percorrer ou obter a visão panorâmica, leia a propriedade `scrollRect` em uma variável separada de Rectangle e altere a propriedade desejada (por exemplo, altere a propriedade `x` da ocorrência de Rectangle para obter a visão panorâmica ou a propriedade `y` para percorrer). Em seguida, atribua novamente essa ocorrência de Rectangle à propriedade `scrollRect` para notificar o objeto de exibição do valor alterado.

Por exemplo, o código a seguir define a área visível de um objeto TextField chamado `bigText` que é muito pequeno para se adaptar nos limites do arquivo SWF. Quando são clicados, os dois botões chamados `up` e `down` chamam funções que fazem com que o conteúdo do objeto TextField se movimente para cima ou para baixo, modificando a propriedade `y` da ocorrência `scrollRect` de Rectangle.

```
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);
```

Como este exemplo ilustra, ao trabalhar com a propriedade `scrollRect` de um objeto de exibição, é melhor especificar que o Flash Player ou o AIR deve armazenar em cache o conteúdo do objeto de exibição como um bitmap, usando a propriedade `cacheAsBitmap`. Ao fazer isso, o Flash Player e o AIR não precisam redesenhar o conteúdo inteiro do objeto de exibição sempre que for movido e, assim, podem usar o bitmap em cache para renderizar a parte necessária diretamente na tela. Para obter detalhes, consulte "Armazenamento em cache de objetos de exibição" na página 185.

Manipulação do tamanho e dimensionamento de objetos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível medir e manipular o tamanho de um objeto de exibição de duas formas, usando as propriedades de dimensão (`width` e `height`) ou as propriedades de escala (`scaleX` e `scaleY`).

Todo objeto de exibição tem uma propriedade `width` e uma propriedade `height`, que são definidas inicialmente como o tamanho do objeto em pixels. É possível ler os valores dessas propriedades para medir o tamanho do objeto de exibição. Você também pode especificar novos valores para alterar o tamanho do objeto, do seguinte modo:

```
// Resize a display object.
square.width = 420;
square.height = 420;

// Determine the radius of a circle display object.
var radius:Number = circle.width / 2;
```

Se você alterar a `altura` ou a `largura` de um objeto de exibição, esse objeto será dimensionado, ampliando ou reduzindo seu conteúdo para ajustá-lo na nova área. Caso o objeto de exibição tenha apenas formas vetoriais, essas formas serão redesenhadas na nova escala, sem prejudicar a qualidade. Todos os elementos gráficos de bitmap do objeto de exibição serão dimensionados, não redesenhados. Assim, por exemplo, uma foto digital cuja largura e altura são aumentadas além das dimensões reais das informações de pixels na imagem será pixelizada, ficando irregular.

Quando as propriedades `width` ou `height` de um objeto de exibição são alteradas, o Flash Player e o AIR também atualizam as propriedades `scaleX` e `scaleY`.

Nota: *Os objetos TextField são uma exceção a este comportamento de dimensionamento. Os campos de texto devem fazer o dimensionamento automático para acomodar textos com quebra e tamanhos de fonte variados, fazendo com que os valores scaleX ou scaleY sejam redefinidos como 1 após o redimensionamento. No entanto, se você ajustar os valores scaleX ou scaleY de um objeto TextField, os valores de largura e altura serão alterados para acomodar os valores de dimensionamento especificados.*

Essas propriedades representam o tamanho relativo do objeto de exibição em comparação com o tamanho original. As propriedades `scaleX` e `scaleY` usam valores fracionários (decimais) para representar porcentagens. Por exemplo, se a `largura` de um objeto de exibição for alterada para ter metade do tamanho original, a propriedade `scaleX` terá o valor `.5`, que indica 50%. Se a altura for dobrada, a propriedade `scaleY` terá o valor `2`, que indica 200%.

```
// 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
```

As alterações de tamanho não são proporcionais. Em outras palavras, e você alterar a `altura` de um quadrado, mas não a `largura`, suas proporções não serão mais as mesmas e o resultado será um retângulo, em vez de um quadrado. Se desejar fazer alterações relacionadas ao tamanho de um objeto de exibição, defina os valores das propriedades `scaleX` e `scaleY` para redimensionar o objeto, em vez de definir as propriedades `width` ou `height`. Por exemplo, este código altera a `largura` do objeto de exibição chamado `square` e, em seguida, altera a escala vertical (`scaleY`) para corresponder à escala horizontal, mantendo o tamanho proporcional do quadrado.

```
// 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;
```

Controle da distorção durante o dimensionamento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Normalmente, quando um objeto de exibição é dimensionado (por exemplo, ampliado na horizontal), a distorção resultante é distribuída uniformemente no objeto, para que cada parte seja ampliada do mesmo modo. Para elementos gráficos e de design, isso é provavelmente o que se espera. No entanto, às vezes é melhor ter controle sobre as partes do objeto de exibição que são ampliadas e as partes que permanecem inalteradas. Um exemplo comum disso é um botão retangular com cantos arredondados. Com o dimensionamento normal, os cantos do botão são ampliados, alterando o raio do canto à medida que o botão é redimensionado.

No entanto, nesse caso seria melhor ter controle sobre o dimensionamento — conseguir designar algumas áreas que devem ser dimensionadas (lados retos e o centro) e outras que não devem (os cantos) — para que o dimensionamento ocorra sem nenhuma distorção visível.

Você pode usar o dimensionamento de 9 fatias (Escala 9) para criar objetos de exibição cujo dimensionamento pode ser controlado. Com o dimensionamento de 9 fatias, o objeto de exibição é dividido em 9 retângulos separados (uma grade 3 x 3, como a grade do jogo da velha). Os retângulos não são necessariamente do mesmo tamanho; você pode desenhar onde as linhas da grade são colocadas. Todo conteúdo que estiver nos retângulos dos quatro cantos (como os cantos arredondados de um botão) não será ampliado ou reduzido quando o objeto de exibição for dimensionado. Os retângulos centrais superior e inferior serão dimensionados na horizontal, não na vertical, enquanto os retângulos centrais esquerdo e direito serão dimensionados na vertical, não na horizontal. O retângulo central será dimensionado tanto na horizontal quanto na vertical.

Com isso em mente, se estiver criando um objeto de exibição e desejar que um determinado conteúdo nunca seja dimensionado, verifique se as linhas divisórias da grade de dimensionamento de 9 fatias estão colocadas de modo que o conteúdo fique em um dos retângulos do canto.

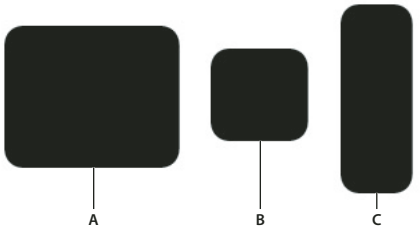
No ActionScript, definir um valor para a propriedade `scale9Grid` de um objeto de exibição ativa o dimensionamento de 9 fatias do objeto e define o tamanho dos retângulos na grade de escala 9 do objeto. Use uma ocorrência da classe Rectangle como valor da propriedade `scale9Grid`, do seguinte modo:

```
myButton.scale9Grid = new Rectangle(32, 27, 71, 64);
```

Os quatro parâmetros do construtor de retângulo são a coordenada x, a coordenada y, a largura e a altura. Neste exemplo, o canto superior esquerdo do retângulo é colocado no ponto x: 32, y: 27 no objeto de exibição chamado `myButton`. O retângulo tem 71 pixels de largura e 64 pixels de altura (de modo que a borda direita está na coordenada 103 do eixo x e a borda inferior está na coordenada 92 do eixo y no objeto de exibição).

A área real contida na região definida pela ocorrência de Rectangle representa o retângulo central da grade de escala 9. Os outros retângulos são calculados pelo Flash Player e AIR, estendendo os lados da ocorrência de Rectangle, conforme mostrado aqui:

Nesse caso, à medida que o botão é dimensionado para cima ou para baixo, os cantos arredondados não são ampliados ou reduzidos, mas as outras áreas se ajustam de acordo com o dimensionamento.



A. *myButton.width = 131;myButton.height = 106;* ***B.*** *myButton.width = 73;myButton.height = 69;* ***C.*** *myButton.width = 54;myButton.height = 141;*

Armazenamento em cache de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Sempre que estiver aumentando o tamanho no Flash, seja para criar um aplicativo ou animações complexas com script, você precisa levar em conta o desempenho e a otimização. O Flash Player e o AIR não otimizam conteúdo que permanece estático (como uma ocorrência de Shape retangular). Desse modo, quando você altera a posição do retângulo, o Flash Player ou o AIR redesenha a ocorrência de Shape inteira.

Você pode armazenar os objetos de exibição especificados em cache para melhorar o desempenho do arquivo SWF. O objeto de exibição é uma *superfície*, basicamente uma versão de bitmap dos dados vetoriais da ocorrência, que são os dados que não devem mudar muito durante o fluxo do arquivo SWF. Portanto, as ocorrências com o cache ativado não são redesenhadas continuamente à medida que o arquivo SWF é reproduzido, o que aumenta a velocidade da renderização.

Nota: *Você pode atualizar os dados vetoriais e, quando isso é feito, a superfície é recriada. Assim, os dados vetoriais armazenados em cache na superfície não precisam permanecer iguais para todo o arquivo SWF.*

Se você definir a propriedade `cacheAsBitmap` do objeto de exibição como `true`, o cache do objeto de exibição será uma representação em bitmap do próprio objeto. O Flash Player ou o AIR criam um objeto de superfície para a ocorrência, que é um bitmap armazenado em cache em vez de dados vetoriais. Se os limites do objeto de exibição forem alterados, a superfície será recriada em vez de ser redimensionada. As superfícies podem ser aninhadas com outras superfícies. A superfície filho copia seu bitmap na superfície pai. Para obter mais informações, consulte "Ativação do armazenamento em cache de bitmaps" na página 187.

A propriedade `opaqueBackground` e a propriedade `scrollRect` da classe DisplayObject estão relacionadas ao armazenamento em cache de bitmaps realizado com a propriedade `cacheAsBitmap`. Embora essas três propriedades sejam independentes entre si, as propriedades `opaqueBackground` e `scrollRect` funcionam melhor quando um objeto é armazenado em cache como um bitmap; você verá a melhora de desempenho das propriedades `opaqueBackground` e `scrollRect` somente quando `cacheAsBitmap` for definido como `true`. Para obter mais informações sobre como percorrer o conteúdo do objeto de exibição, consulte "Visão panorâmica e rolagem de objetos de exibição" na página 181. Para obter mais informações sobre como configurar um plano de fundo opaco, consulte "Definição de uma cor de fundo opaca" na página 188.

Para obter informações sobre o mascaramento do canal alfa, que requer a definição da propriedade `cacheAsBitmap` como `true`, consulte "Mascaramento de objetos de exibição" na página 193.

Quando ativar o armazenamento em cache

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A ativação do cache para um objeto de exibição cria uma superfície, o que tem diversas vantagens, como ajudar na renderização rápida de animações vetoriais complexas. Existem diversos cenários nos quais é necessário ativar o cache. Você talvez pense que sempre que o cache é ativado, o desempenho dos arquivos SWF melhora; no entanto, existem situações nas quais a ativação do cache não melhora o desempenho ou pode inclusive piorá-lo. Esta seção descreve cenários nos quais o cache deve ser usado e quando objetos de exibição regulares devem ser usados.

O desempenho global de dados em cache depende da complexidade dos dados vetoriais das ocorrências, quanto foram mudados os dados e se foi ou não definida a propriedade `opaqueBackground`. Se você estiver mudando regiões pequenas, a diferença entre o uso de superfície e o uso de dados vetoriais pode ser desprezível. Teste o seu trabalho das duas formas, antes de implantar o aplicativo.

Quando usar o armazenamento em cache de bitmaps

A seguir, alguns casos comuns nos quais podem ser vistos benefícios significativos quando se ativa o armazenamento em cache de bitmaps.

- Imagem de fundo complexa: um aplicativo que contém uma imagem de fundo complexa e detalhada dos dados vetoriais (talvez uma imagem na qual o comando Traçar bitmap tenha sido aplicado ou a arte final criada no Adobe Illustrator®). Você pode animar caracteres no plano de fundo, o que deixa a animação mais lenta porque o plano de fundo precisa gerar regularmente os dados vetoriais mais uma vez. Para melhorar o desempenho, você pode definir a propriedade `opaqueBackground` do objeto de exibição de fundo como `true`. O fundo é renderizado como um bitmap e pode se redesenhado rapidamente, de modo que a execução da animação seja muito mais rápida.
- Rolagem de campo de texto: um aplicativo que exibe uma grande quantidade de texto na rolagem do campo de texto. Você pode colocar o campo de texto em um objeto de exibição definido como rolável com limites de rolagem (a propriedade `scrollRect`). Isso agiliza a rolagem de pixel para a ocorrência especificada. Quando o usuário rola a ocorrência do objeto de exibição, o Flash Player ou o AIR move os pixels rolados para cima e gera a região recém-exposta, em vez de gerar novamente todo o campo de texto.

- Sistema de janelas: um aplicativo com um sistema complexo de janelas sobrepostas. Cada janela pode ser aberta ou fechada (por exemplo, as janelas de navegador da web). Se você marcar cada janela como uma superfície (definindo a propriedade `cacheAsBitmap` como `true`), cada janela é isolada e colocada em cache. Os usuários podem arrastar as janelas de modo que se sobreponham, e as janelas não precisam gerar novamente o conteúdo vetorial.
- Mascaramento do canal alfa: ao usar o mascaramento do canal alfa, você deve definir a propriedade `cacheAsBitmap` como `true`. Para obter mais informações, consulte "Mascaramento de objetos de exibição" na página 193.

A ativação do cache de bitmaps em todos esses cenários melhora a resposta e a interatividade do aplicativo, otimizando os gráficos vetoriais.

Além disso, sempre que um filtro é aplicado em um objeto de exibição, `cacheAsBitmap` é definido automaticamente como `true`, mesmo que esteja explicitamente definido como `false`. Se todos os filtros forem desativados do objeto de exibição, a propriedade `cacheAsBitmap` retornará ao valor definido pela última vez.

Quando evitar o uso do armazenamento em cache de bitmaps

Usando este recurso nas circunstâncias erradas pode prejudicar o desempenho do arquivo SWF. Ao usar o cache de bitmaps, lembre-se das seguintes orientações:

- Não use em exagero superfícies (objetos de exibição com o cache ativado). Cada superfície usa mais memória do que um objeto de exibição normal, o que indica que você deve ativar as superfícies apenas quando precisar melhorar o desempenho da renderização.

  Um bitmap em cache pode usar significativamente mais memória do que um objeto de exibição normal. Por exemplo, se uma ocorrência de Sprite no palco tem 250 pixels por 250 pixels de tamanho, pode usar 250 KB em vez de 1 KB em cache quando for uma ocorrência normal de Sprite (não armazenada em cache).
- Evite o zoom em superfícies em cache. Se usar exageradamente cache de bitmaps, é consumida uma grande quantidade de memória (veja observação anterior) se fizer o zoom do conteúdo.
- Use superfícies para ocorrências de objeto de exibição em grande parte estáticas (sem animação). Você pode arrastar ou mover a ocorrência, mas o conteúdo da ocorrência não deve ser animado ou mudado muito. A animação ou alteração do conteúdo tem maior probabilidade de acontecer com uma ocorrência de MovieClip que contém animação ou uma ocorrência de Video. Por exemplo, se você girar ou transformar uma ocorrência, ela muda entre a superfície e os dados vetoriais, o que é difícil de processar e afeta negativamente o arquivo SWF.
- Se misturar superfícies com dados vetoriais, aumenta a quantidade de processamento a ser feita pelo Flash Player e pelo AIR (e algumas vezes o computador). Agrupe as superfícies o máximo possível, por exemplo, quando criar aplicativos em janelas.
- Não armazene em cache objetos cujas imagens gráficas sejam alteradas com frequência. Toda vez que você dimensiona, inclina, gira o objeto de exibição, altera a transformação de cor ou do alfa, move objetos de exibição filho ou desenha usando a propriedade graphics, o cache de bitmap é redesenhado. Se isso acontecer em cada quadro, o tempo de execução precisa desenhar o objeto em um bitmap e, em seguida, copiar esse bitmap no palco, o que resulta em trabalho extra, quando comparado com o simples desenho no palco do objeto que não está em cache. A escolha em termos de desempenho entre cache e frequência de atualização depende da complexidade e do tamanho do objeto de exibição e pode ser determinada somente pelo teste do conteúdo específico.

Ativação do armazenamento em cache de bitmaps

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para ativar o armazenamento em cache de bitmaps para um objeto de exibição, defina a propriedade `cacheAsBitmap` como `true`:

```
mySprite.cacheAsBitmap = true;
```

Depois de definir a propriedade `cacheAsBitmap` como `true`, você deve perceber que o objeto de exibição realiza o encaixe de pixels automaticamente em coordenadas inteiras. Ao testar o arquivo SWF, você verá que qualquer animação executada em uma imagem vetorial complexa é processada muito mais rápido.

Uma superfície (bitmap em cache) não será criada, mesmo que `cacheAsBitmap` esteja definido como `true`, se ocorrer uma ou mais das seguintes situações:

- O bitmap é muito maior do que 2880 pixels de altura ou largura.
- O bitmap não consegue ser alocado (produzindo erro de falta de memória).

Matrizes de transformação de bitmap armazenado em cache

Adobe AIR 2.0 e posterior (perfil móvel)

Em aplicativos AIR para dispositivos móveis, você deve definir a propriedade `cacheAsBitmapMatrix` sempre que definir a propriedade `cacheAsBitmap`. A definição desta propriedade permite aplicar um intervalo mais amplo de transformações ao objeto de exibição sem acionamento de uma nova renderização.

```
mySprite.cacheAsBitmap = true;
mySprite.cacheAsBitmapMatrix = new Matrix();
```

Quando você define esta matriz adequadamente, pode aplicar a seguinte transformação adicional ao objeto de exibição sem voltar a armazenar o objeto em cache:

- Mover ou converter sem encaixe de pixel
- Girar
- Dimensionamento
- Inclinar
- Alterar o alfa (entre 0% e 100% de transparência)

Essas transformações são aplicadas diretamente ao bitmap no cache.

Definição de uma cor de fundo opaca

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode definir um plano de fundo opaco para um objeto de exibição. Por exemplo, quando o SWF tem um plano de fundo que contém elementos vetoriais complexos, você pode definir a propriedade `opaqueBackground` como uma cor especificada (normalmente a mesma cor do palco). A cor é especificada como um número (em geral, um valor de cor hexadecimal). O plano de fundo é tratado como um bitmap, o que ajuda a otimizar o desempenho.

Quando você define `cacheAsBitmap` como `true` e também define a propriedade `opaqueBackground` como uma cor especificada, a propriedade `opaqueBackground` permite que o bitmap interno seja opaco e renderizado mais rapidamente. Se `cacheAsBitmap` não for definido como `true`, a propriedade `opaqueBackground` adicionará uma forma vetorial quadrada opaca ao plano de fundo do objeto de exibição. Isso não cria um bitmap automaticamente.

O exemplo a seguir mostra como definir o plano de fundo de um objeto de exibição para otimizar o desempenho:

```
myShape.cacheAsBitmap = true;
myShape.opaqueBackground = 0xFF0000;
```

Nesse caso, a cor de fundo da forma chamada `myShape` é definida como vermelha (`0xFF0000`). Supondo que a ocorrência de Shape contém um desenho de um triângulo verde, em um palco com fundo branco, seria mostrado um triângulo verde com vermelho no espaço vazio da caixa delimitadora da ocorrência de Shape (o retângulo que envolve a forma por completo).

Obviamente, isso faria mais sentido se fosse usado com um palco com fundo vermelho sólido. Em outro fundo colorido, essa cor seria especificada. Por exemplo, em um SWF com fundo branco, a propriedade `opaqueBackground` provavelmente seria definida como `0xFFFFFF`, ou branco puro.

Aplicação de modos de mesclagem

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os modos de mesclagem envolvem a combinação das cores de uma imagem (a imagem base) com as cores de outra imagem (a imagem de mesclagem) para produzir uma terceira imagem; a imagem resultante é aquela realmente exibida na tela. Cada valor de pixel em uma imagem é processado com o valor de pixel correspondente da outra imagem para produzir um valor de pixel para a mesma posição no resultado.

Todos os objetos de exibição têm uma propriedade `blendMode` que pode ser definida como um dos seguintes modos de mesclagem. Esses modos são constantes definidas na classe BlendMode. Se preferir, você pode usar os valores de String (entre parênteses) que são os reais valores das constantes.

- `BlendMode.ADD` (`"add"`): normalmente usado para criar um efeito animado de dissolução de iluminação entre duas imagens.
- `BlendMode.ALPHA` (`"alpha"`): normalmente usado para aplicar a transparência do primeiro plano no plano de fundo. (Sem suporte na renderização pela GPU.)
- `BlendMode.DARKEN` (`"darken"`): normalmente usado para sobrepor tipos. (Sem suporte na renderização pela GPU.)
- `BlendMode.DIFFERENCE` (`"difference"`): normalmente usado para criar cores mais vibrantes.
- `BlendMode.ERASE` (`"erase"`): normalmente usado para cortar (apagar) parte do plano de fundo usando o alfa do primeiro plano. (Sem suporte na renderização pela GPU.)
- `BlendMode.HARDLIGHT` (`"hardlight"`): normalmente usado para criar efeitos de sombra. (Sem suporte na renderização pela GPU.)
- `BlendMode.INVERT` (`"invert"`): usado para inverter o plano de fundo.
- `BlendMode.LAYER` (`"layer"`): usado para forçar a criação de um buffer temporário para pré-composição de um objeto de exibição específico. (Sem suporte na renderização pela GPU.)
- `BlendMode.LIGHTEN` (`"lighten"`): normalmente usado para sobrepor tipos. (Sem suporte na renderização pela GPU.)
- `BlendMode.MULTIPLY` (`"multiply"`): normalmente usado para criar sombras e efeitos de profundidade.
- `BlendMode.NORMAL` (`"normal"`): usado para especificar que os valores de pixel da imagem de mesclagem substituem os da imagem base.
- `BlendMode.OVERLAY` (`"overlay"`): normalmente usado para criar efeitos de sombra. (Sem suporte na renderização pela GPU.)

- `BlendMode.SCREEN` (`"screen"`): normalmente usado para criar realces e manchas de luz.
- `BlendMode.SHADER` (`"shader"`): usado para especificar que um sombreador Pixel Bender é usado para criar um efeito de mesclagem personalizado. Para obter mais informações sobre como usar sombreadores, consulte "Trabalho com sombreadores Pixel Bender" na página 308. (Sem suporte na renderização pela GPU.)
- `BlendMode.SUBTRACT` (`"subtract"`): normalmente usado para criar um efeito animado de dissolução de escurecimento entre duas imagens.

Ajuste das cores de DisplayObject

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar os métodos da classe ColorTransform (flash.geom.ColorTransform) para ajustar a cor de um objeto de exibição. Cada objeto de exibição tem uma propriedade `transform`, que é uma ocorrência da classe Transform, e contém informações sobre várias transformações que são aplicadas no objeto de exibição (como rotação, alterações na escala ou posição e assim por diante). Além de informações sobre transformações geométricas, a classe Transform também inclui uma propriedade `colorTransform`, que é uma ocorrência da classe ColorTransform e fornece acesso para fazer ajustes de cor no objeto de exibição. Para acessar as informações de transformação de cor de um objeto de exibição, você pode usar um código como esse:

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;
```

Depois de criar uma ocorrência de ColorTransform, você pode ler os valores de propriedade para descobrir quais transformações de cor já foram aplicadas ou definir esses valores para alterar cores no objeto de exibição. Para atualizar o objeto de exibição depois de fazer alterações, atribua novamente a ocorrência de ColorTransform à propriedade `transform.colorTransform`.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;

// Make some color transformations here.

// Commit the change.
myDisplayObject.transform.colorTransform = colorInfo;
```

Definição de valores de cor com código

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A propriedade `color` da classe ColorTransform pode ser usada para atribuir um valor de cor RGB (vermelho, verde e azul) específico ao objeto de exibição. O exemplo a seguir usa a propriedade `color` para alterar a cor do objeto de exibição chamado `square` como azul quando o usuário clicar no botão `blueBtn`:

```
// 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);
```

Observe que, ao alterar a cor de um objeto de exibição usando a propriedade `color`, a cor do objeto inteiro é alterada, independentemente de o objeto ter várias cores anteriormente. Por exemplo, se houver um objeto de exibição que contém um círculo verde com texto preto na parte superior, definir a propriedade `color` da ocorrência de ColorTransform associada desse objeto como uma sombra vermelha faz com que o objeto inteiro, círculo e texto, fique vermelho (de modo que o texto não será mais diferenciado do restante do objeto).

Alteração de efeitos de brilho e cor com código

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Digamos que você tenha um objeto de exibição com várias cores (por exemplo, uma foto digital) e não queira colorir o objeto inteiro novamente; você quer apenas ajustar a cor de um objeto de exibição com base nas cores existentes. Nesse cenário, a classe ColorTransform inclui uma série de propriedades de multiplicador e deslocamento que podem ser usadas para fazer esse tipo de ajuste. As propriedades de multiplicador, chamadas `redMultiplier`, `greenMultiplier`, `blueMultiplier` e `alphaMultiplier`, funcionam como filtros fotográficos coloridos (ou óculos de sol coloridos), intensificando ou ofuscando algumas cores no objeto de exibição. As propriedades de deslocamento (`redOffset`, `greenOffset`, `blueOffset` e `alphaOffset`) podem ser usadas para aumentar a quantidade de uma determinada cor no objeto ou para especificar o valor mínimo que uma cor específica pode ter.

Essas propriedades de multiplicador e deslocamento são idênticas às configurações de cor avançadas que estão disponíveis para símbolos de clipe de filme na ferramenta de criação do Flash quando você escolhe Avançado no menu pop-up Cor no Inspetor de propriedades.

O código a seguir carrega uma imagem JPEG e aplica uma transformação de cor nela, ajustando os canais vermelho e verde à medida que o ponteiro do mouse se move ao longo dos eixos x e y. Nesse caso, como nenhum valor de deslocamento foi especificado, o valor de cada canal de cor exibido na tela será uma porcentagem do valor de cor original na imagem - a maior parte de vermelho ou verde exibida em um pixel é a quantidade original de vermelho ou verde nesse pixel.

```
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);
```

Rotação de objetos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os objetos de exibição podem ser girados com a propriedade `rotation`. Você pode ler esse valor para saber se um objeto foi girado ou, para girar o objeto, defina essa propriedade como um número (em graus) que representa o valor de rotação a ser aplicado no objeto. Por exemplo, essa linha do código gira o objeto chamado `square` 45 graus (1/8 de uma revolução completa):

```
square.rotation = 45;
```

Se preferir, gire o objeto de exibição usando uma matriz de transformação, conforme descrito em "Trabalho com geometria" na página 214.

Desaparecimento de objetos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode controlar a transparência de um objeto de exibição para deixá-lo parcial ou completamente transparente ou alterar a transparência para realçar ou ofuscar o objeto. A propriedade `alpha` da classe DisplayObject define a transparência (ou, mais precisamente, a opacidade) de um objeto de exibição. A propriedade `alpha` pode ser definida como qualquer valor entre 0 e 1, onde 0 é completamente transparente e 1 é completamente opaco. Por exemplo, essas linhas de código deixam o objeto chamado `myBall` parcialmente transparente (50%) quando é clicado com o mouse:

```
function fadeBall(event:MouseEvent):void
{
    myBall.alpha = .5;
}
myBall.addEventListener(MouseEvent.CLICK, fadeBall);
```

Você também pode alterar a transparência de um objeto de exibição usando os ajustes de cor disponíveis pela classe ColorTransform. Para obter mais informações, consulte "Ajuste das cores de DisplayObject" na página 190.

Mascaramento de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar um objeto de exibição como uma máscara para criar um orifício por meio do qual o conteúdo de outro objeto de exibição é visualizado.

Definição de uma máscara

Para indicar que um objeto de exibição será a máscara de outro objeto, defina o objeto de máscara como a propriedade `mask` do objeto de exibição a ser mascarado:

```
// Make the object maskSprite be a mask for the object mySprite.
mySprite.mask = maskSprite;
```

O objeto de exibição mascarado é revelado em todas as áreas opacas (não transparentes) do objeto de exibição que age como a máscara. Por exemplo, o código a seguir cria uma ocorrência de Shape que contém um quadrado vermelho de 100 x 100 pixels e uma ocorrência de Sprite que contém um círculo azul com raio de 25 pixels. Assim que é clicado, o círculo é definido como a máscara do quadrado para que a única parte exibida do quadrado seja a parte coberta pela parte sólida do círculo. Em outras palavras, somente um círculo vermelho ficará visível.

```
// 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);
```

O objeto de exibição que atua como máscara pode ser arrastado, animado, redimensionado dinamicamente e pode usar formas separadas em uma única máscara. O objeto de exibição de máscara não precisa ser necessariamente adicionado à lista de exibição. No entanto, se desejar que o objeto de máscara seja dimensionado quando o palco for dimensionado ou se desejar permitir a interação do usuário com a máscara (como as operações de arrastar e redimensionar controladas pelo usuário), o objeto de máscara deve ser adicionado à lista de exibição. O índice z real (ordem da frente para trás) dos objetos de exibição não importam, desde que o objeto de máscara seja adicionado à lista de exibição. O objeto de máscara será exibido na tela apenas como uma máscara. Se o objeto de máscara for uma ocorrência de MovieClip com vários quadros, todos os quadros da linha do tempo desse objeto serão reproduzidos, do mesmo modo como se não estivesse agindo como máscara. Para remover uma máscara, defina a propriedade `mask` como `null`:

```
// remove the mask from mySprite
mySprite.mask = null;
```

Você não pode usar uma máscara para mascarar outra. A propriedade `alpha` de um objeto de exibição de máscara não pode ser definida. Somente os preenchimentos são usados em um objeto de exibição usado como máscara; os traçados são ignorados.

AIR 2

Se um objeto de exibição com máscara for armazenado em cache pela definição das propriedades `cacheAsBitmap` e `cacheAsBitmapMatrix`, a máscara deve ser um filho do objeto de exibição com máscara. Da mesma forma, se o objeto de exibição com máscara for descendente de um contêiner de objetos de exibição em cache, a máscara e o objeto de exibição devem ser descendentes deste contêiner. Se o objeto com máscara for descendente de mais de um contêiner de objetos de exibição em cache, a máscara deve ser descendente do recipiente em cache mais próximo do objeto com máscara na lista de exibição.

Sobre o mascaramento de fontes de dispositivo

Você pode usar um objeto de exibição para mascarar o texto que está definido em uma fonte de dispositivo. Quando um objeto de exibição é usado para mascarar o texto definido em uma fonte de dispositivo, a caixa delimitadora retangular da máscara é usada como a forma de mascaramento. Isso significa que, se você criar uma máscara não retangular de objeto de exibição para o texto da fonte de dispositivo, a máscara exibida no arquivo SWF terá a forma da caixa delimitadora retangular da máscara, não a forma da máscara propriamente dita.

Mascaramento do canal alfa

O mascaramento do canal alfa é permitido se a máscara e os objetos de exibição mascarados usarem o cache de bitmaps, conforme mostrado aqui:

```
// maskShape is a Shape instance which includes a gradient fill.
mySprite.cacheAsBitmap = true;
maskShape.cacheAsBitmap = true;
mySprite.mask = maskShape;
```

Por exemplo, o mascaramento do canal alfa usa um filtro no objeto de máscara diferente do filtro que é aplicado no objeto de exibição mascarado.

No exemplo a seguir, um arquivo de imagem externo é carregado no palco. Essa imagem (ou, mais precisamente, a ocorrência de Loader na qual é carregada) será o objeto de exibição mascarado. Um elemento oval de gradiente (centro preto sólido que fica transparente nas bordas) é desenhado na imagem; esta será a máscara alfa. Os dois objetos de exibição têm o cache de bitmaps ativado. O elemento oval é definido como uma máscara para a imagem e pode ser arrastado.

```
// 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);
```

Animação de objetos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Animação é o processo de fazer algo se movimentar ou, também, de fazer algo mudar com o passar do tempo. A animação com script é uma parte fundamental dos jogos de vídeo e normalmente é usada para adicionar dicas úteis de interação a outros aplicativos.

A ideia básica por trás da animação por script é de que uma alteração deve ocorrer e essa alteração precisa ser dividida em incrementos com o passar do tempo. É fácil fazer algo se repetir no ActionScript, usando uma instrução de consulta comum. No entanto, uma consulta será executada em todas as iterações antes da atualização da exibição. Para criar animação com script, você precisa gravar um ActionScript que execute alguma ação repetidas vezes com o passar do tempo e também atualize a tela sempre que essa ação for executada.

Por exemplo, imagine que você quer criar uma animação simples, como uma bola que percorre a tela. O ActionScript inclui um mecanismo fácil que permite acompanhar a passagem do tempo e atualizar a tela conforme necessário, ou seja, você pode gravar o código que move a bola um pouco por vez até atingir o destino. Após cada movimentação, a tela é atualizada e o usuário pode visualizar o movimento no palco.

Do ponto de vista prático, faz sentido sincronizar a animação com script com a taxa de quadros do arquivo SWF (em outras palavras, fazer uma alteração de animação sempre que um novo quadro for exibido), pois tal procedimento define a frequência de atualização de tela do Flash Player ou do AIR. Cada objeto de exibição tem um evento `enterFrame` que é enviado de acordo com a taxa de quadros do arquivo SWF - um evento por quadro. A maioria dos desenvolvedores que cria animações com script usa o evento `enterFrame` para criar ações que se repetem com o passar do tempo. Você pode gravar um código que ouve o evento `enterFrame`, movendo a bola animada um pouco em cada quadro e, à medida que a tela é atualizada (cada quadro), a bola seria redesenhada em seu novo local, criando o movimento.

Nota: *Uma ação que se repete com o passar do tempo também pode ser criada com a classe Timer. Uma ocorrência de Timer aciona uma notificação de evento sempre que um período especificado passa. Você poderia gravar um código que executa a animação manipulando os eventos de tempo da classe Timer, definindo um intervalo de tempo pequeno (alguma fração de um segundo). Para obter mais informações sobre como usar a classe Timer, consulte "Controle de intervalos de tempo" na página 4.*

No exemplo a seguir, uma ocorrência circular de Sprite, chamada `circle`, é criada no palco. Quando o usuário clica no círculo, uma sequência de animação com script é iniciada, fazendo com que o `círculo` desapareça (a propriedade `alpha` é diminuída) até ficar completamente transparente:

```
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);
```

Quando o usuário clica no círculo, a função `fadeCircle()` é inscrita como ouvinte do evento `enterFrame`, indicando que começará a ser chamada uma vez por quadro. Essa função desbota o `círculo` alterando sua propriedade `alpha` para que, uma vez por quadro, o `alfa` do círculo diminua 0,05 (5%) e a tela seja atualizada. Eventualmente, quando o valor de `alpha` é 0 (`círculo` completamente transparente), a função `fadeCircle()` é removida como ouvinte de eventos, encerrando a animação.

O mesmo código poderia ser usado, por exemplo, para criar um movimento animado em vez de desbotar o objeto. Substituindo uma propriedade diferente para `alpha` na função que é um ouvinte de eventos `enterFrame`, essa propriedade será animada. Por exemplo, alterar esta linha

```
circle.alpha -= .05;
```

para este código

```
circle.x += 5;
```

animará a propriedade `x`, fazendo com que o círculo se mova para a direita no palco. A condição que encerra a animação poderia ser alterada para finalizar a animação (isto é, cancelar a inscrição do ouvinte `enterFrame`) quando a coordenada x desejada for atingida.

Orientação do palco

AIR 2.0 e posterior

Normalmente, dispositivos móveis reorientam a interface do usuário para manter a exibição vertical quando o usuário gira o dispositivo. Se você ativar a orientação automática em seu aplicativo, o dispositivo manterá o visor corretamente orientado, mas cabe a você se certificar de que a aparência do conteúdo esteja OK quando a proporção do palco for alterada. Se você desativar a auto-orientação, a exibição do dispositivo permanecerá fixa, a menos que você altere a orientação manualmente.

Os aplicativos AIR são executados em diversos dispositivos e sistemas operacionais diferentes. O comportamento subjacente da orientação pode variar dependendo do sistema operacional e, até mesmo, dos diferentes dispositivos com o mesmo sistema operacional. Uma simples estratégia de design, que funciona bem em todos os dispositivos e sistemas operacionais, é ativar a orientação automática e ouvir eventos `resize` do palco para determinar quando é preciso atualizar o layout do aplicativo.

Como alternativa, se seu aplicativo oferecer suporte somente à proporção retrato ou paisagem, você pode desativar a orientação automática e definir a proporção com suporte no descritor do aplicativo do AIR. Esta estratégia de design proporciona um comportamento consistente e seleciona a "melhor" orientação para a proporção selecionada. Por exemplo, se você especificar a proporção de paisagem, a orientação escolhida será apropriada para dispositivos com teclados slide-out em modo de paisagem.

Obtenção da orientação e proporção atuais do palco

A orientação é informada em relação à posição normal do dispositivo. Na maioria dos dispositivos, há uma posição vertical clara. Essa posição é considerada a orientação *padrão*. As outras três orientações possíveis são: *girado para a esquerda*, *girado para a direita* e *de ponta cabeça*. A classe StageOrientation define constantes de string a serem usadas na definição ou comparação de valores de orientação.

A classe Stage define duas propriedades que indicam a orientação:

- Stage.deviceOrientation – Indica a orientação física do dispositivo em relação à posição padrão.

  Nota: *A propriedade deviceOrientation nem sempre está disponível quando o aplicativo é inicializado ou quando o dispositivo está deitado sobre uma superfície. Nesses casos, a orientação do dispositivo é indicada como desconhecida.*

- Stage.orientation – Indica a orientação do palco em relação à posição padrão. Quando a orientação automática é ativada, o palco gira na direção oposta à do dispositivo para permanecer vertical. Assim, as posições direita e esquerda indicadas pela propriedade `orientation` são opostas àquelas indicadas pela propriedade `deviceOrientation`. Por exemplo, quando `deviceRotation` indica *girado para a direita*, `orientation` indica *girado para a esquerda*.

A proporção do palco pode ser derivada pela simples comparação da largura e altura atuais do palco:

```
var aspect:String = this.stage.stageWidth >= this.stage.stageHeight ?
StageAspectRatio.LANDSCAPE : StageAspectRatio.PORTRAIT;
```

Orientação automática

Quando a orientação automática está ativa e um usuário gira o dispositivo, o sistema operacional reorienta toda a interface do usuário, inclusive a barra de tarefas e o seu aplicativo. Como resultado, a proporção do palco se altera de retrato para paisagem ou vice-versa. Quando a proporção é alterada, as dimensões do palco também são alteradas.

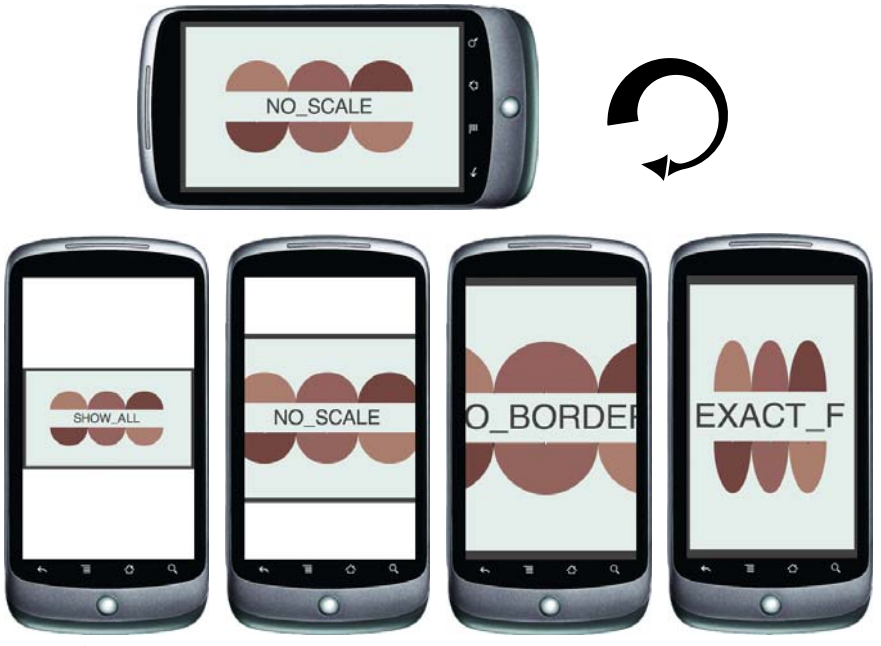
Ative ou desative a orientação automática em tempo de execução definido a propriedade `autoOrients` do palco como `true` ou `false`. Também é possível definir o valor inicial dessa propriedade no descritor do aplicativo do AIR com o elemento `<autoOrients>`. (Observe que antes do AIR 2.6, `autoOrients` é uma propriedade somente leitura que pode ser definida somente no descritor do aplicativo.)

Se você especificar uma taxa de proporção de paisagem ou retrato e também habilitar a orientação automática, o AIR restringe a orientação automática à taxa de proporção especificada.

Alterações das dimensões do palco

Quando as dimensões do palco são alteradas, o conteúdo do palco é dimensionado e reposicionado, conforme especificado pelas propriedades `scaleMode` e `align` do objeto Stage. Na maioria dos casos, depender do comportamento automático fornecido pela configuração de `scaleMode` de Stage não produz bons resultados. Em vez disso, refaça o layout ou redesenhe seus elementos gráficos e componentes de modo a oferecer suporte a mais de uma proporção. (Proporcionar lógica de layout flexível também significa que seu aplicativo funcionará melhor em diferentes dispositivos com diferentes tamanhos e proporções de tela.)

A ilustração a seguir demonstra os efeitos de diferentes configurações de `scaleMode` quando um dispositivo móvel típico é girado:



Rotação da proporção paisagem para retrato

A ilustração demonstra o comportamento de dimensionamento que ocorre na rotação da proporção paisagem para retrato com diferentes modos de escala. Girar de retrato para paisagem causa um conjunto semelhante de efeitos.

Eventos de mudança de orientação

O objeto Stage despacha dois tipos de evento que você pode usar para detectar e reagir a mudanças de orientação. Ambos os eventos `resize` e `orientationChange` do palco são despachados quando a orientação automática é ativada.

O evento *resize* é o melhor evento a ser usado quando você está dependendo da orientação automática para manter o visor na vertical. Quando o palco despachar um evento `resize`, recrie o layout ou redesenhe seu conteúdo, conforme necessário. O evento `resize` é despachado somente quando o modo de escala do palco está definido como `noScale`.

O evento `orientationChange` também pode ser usado para detectar mudanças de orientação. O evento `orientationChange` é despachado somente quando a orientação automática está ativada.

Nota: *Em algumas plataformas móveis, o palco despacha um evento `orientationChanging` cancelável antes de despachar os eventos resize ou orientationChange. Visto que não há suporte para o evento em todas as plataformas, evite depender dele.*

Orientação manual

AIR 2.6 e posterior

Você pode controlar a orientação do palco usando o método de Palco `setOrientation()` ou `setAspectRatio()`.

Definindo a orientação do palco

Você pode definir a orientação do palco no tempo de execução usando o método `setOrientation()` do objeto Stage. Use as constantes de sequências de caracteres definidas pela classe StageOrientation para especificar a orientação desejada:

```
this.stage.setOrientation( StageOrientation.ROTATED_RIGHT );
```

Nem todos os dispositivos ou sistemas operacionais suportam todas as orientações possíveis. Por exemplo, o Android 2.2 não suporta a escolha programada da orientação com giro para a esquerda nos dispositivos que têm a orientação de retrato como padrão, assim como não suporta de nenhuma forma a orientação de cima para baixo. A propriedade `supportedOrientations` do palco fornece uma lista de orientações que podem ser transferidas para o método `setOrientation()`:

```
var orientations:Vector.<String> = this.stage.supportedOrientations;
for each( var orientation:String in orientations )
{
    trace( orientation );
}
```

Definindo a proporção do palco

Se a proporção do palco for sua preocupação principal, você poderá defini-la como retrato ou paisagem. Você pode definir a proporção no descritor do aplicativo do AIR ou no tempo de execução, usando o método `setAspectRatio()`:

```
this.stage.setAspectRatio( StageAspectRatio.LANDSCAPE );
```

O tempo de execução escolhe uma das duas orientações possíveis para a proporção especificada. Essa pode não corresponder à orientação do dispositivo atual. Por exemplo, a orientação padrão é escolhida em vez da orientação de cabeça para baixo (AIR 3.2 e anterior) e a orientação apropriada para o teclado deslizante é escolhida em vez da orientação oposta.

(AIR 3.3 e superior) Começando com o AIR 3.3 (SWF versão 16), você também pode usar a constante `StageAspectRatio.ANY`. Se `Stage.autoOrients` estiver definido como `true` e você chamar `setAspectRatio(StageAspectRatio.ANY)`, seu aplicativo tem o recurso de reorientar todas as orientações (paisagem à esquerda, paisagem à direita, retrato e retrato de cabeça para baixo). Também novo no AIR 3.3, o aspecto de proporção é persistente e a rotação adicional do dispositivo é restrita à orientação especificada.

Exemplo: Definição da orientação do palco para corresponder à orientação do dispositivo

O exemplo a seguir ilustra uma função que atualiza a orientação do palco para corresponder à orientação atual do dispositivo. A propriedade `deviceOrientation` do palco indica a orientação física do dispositivo, mesmo quando a auto-orientação está desativada.

```
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
    }
}
```

A mudança de orientação é assíncrona. Você pode ouvir o evento `orientationChange` despachado pelo palco para detectar a conclusão da mudança. Se uma orientação não for suportada num dispositivo, a chamada `setOrientation()` falhará sem emitir um erro.

Carregamento dinâmico do conteúdo da exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode carregar qualquer uma dos seguintes ativos de exibição externos em um aplicativo ActionScript 3.0:

- Um arquivo SWF criado no ActionScript 3.0 - Esse arquivo pode ser uma classe Sprite, MovieClip ou qualquer classe que estende Sprite. Nos aplicativos do AIR que rodam no iOS, somente os arquivos SWF que não contêm código de byte do ActionScript podem ser carregados. Isso significa que os arquivos SWF que contêm dados incorporados, tais como imagens e som, podem ser carregados, mas não os arquivos SWF que contêm código executável.
- Um arquivo de imagem - Isso inclui arquivos JPG, PNG e GIF.
- Um arquivo SWF AVM1 - Arquivo SWF gravado no ActionScript 1.0 ou 2.0. (não suportado em aplicativos móveis do AIR)

Carregue esses ativos usando a classe Loader.

Carregamento de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os objetos Loader são usados para carregar arquivos SWF e de imagem em um aplicativo. A classe Loader é uma subclasse de DisplayObjectContainer. Um objeto Loader pode conter apenas um objeto de exibição filho na lista de exibição, o objeto que representa o arquivo SWF ou de imagem carregado. Quando você adiciona um objeto Loader à lista de exibição, como no código a seguir, também pode adicionar o objeto filho carregado à lista de exibição após o carregamento:

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

Assim que o arquivo SWF ou imagem é carregado, você pode mover o objeto de exibição carregado para outro contêiner, como o objeto `container` DisplayObjectContainer neste exemplo:

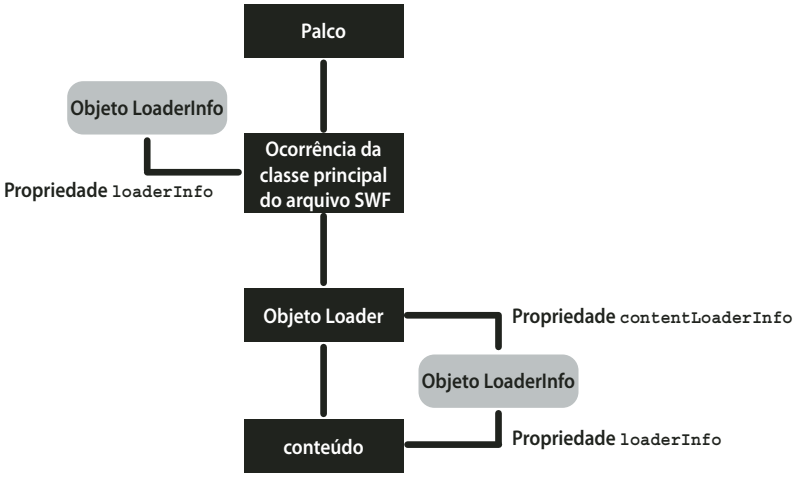
```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
    container.addChild(pictLdr.content);
}
```

Monitoramento do progresso do carregamento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Assim que o arquivo começa a ser carregado, um objeto LoaderInfo é criado. Um objeto LoaderInfo fornece informações como o progresso do carregamento, os URLs do carregador e do conteúdo carregado, o número de bytes totais para a mídia e a altura e largura nominais da mídia. O objeto LoaderInfo também envia eventos para o monitoramento do progresso do carregamento.

O diagrama a seguir mostra os diferentes usos do objeto LoaderInfo - para a ocorrência da classe principal do arquivo SWF, para um objeto Loader e para um objeto carregado por Loader:



O objeto LoaderInfo pode ser acessado como uma propriedade do objeto Loader e do objeto de exibição carregado. Assim que o carregamento começa, o objeto LoaderInfo pode ser acessado por meio da propriedade `contentLoaderInfo` do objeto Loader. Quando o carregamento do objeto de exibição termina, o objeto LoaderInfo também pode ser acessado como uma propriedade do objeto de exibição carregado pela propriedade `loaderInfo`. A propriedade `loaderInfo` do objeto de exibição carregado refere-se ao mesmo objeto LoaderInfo da propriedade `contentLoaderInfo` do objeto Loader. Em outras palavras, um objeto LoaderInfo é compartilhado entre um objeto carregado e o objeto Loader que o carregou (entre o carregador e o carregado).

Para acessar as propriedades do conteúdo carregado, adicione um ouvinte de eventos ao objeto LoaderInfo, assim como no código a seguir:

```
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;
}
```

Para obter mais informações, consulte “Manipulação de eventos” na página 126.

Especificação do contexto do carregamento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando você carrega um arquivo externo no Flash Player ou no AIR com o método `load()` ou `loadBytes()` da classe Loader, pode especificar, se desejar, um parâmetro `context`. Este parâmetros é um objeto LoaderContext.

A classe LoaderContext inclui três propriedades que permitem definir o contexto de como o conteúdo carregado pode ser usado:

- `checkPolicyFile`: Use essa propriedade apenas ao carregar um arquivo de imagem (não um arquivo SWF). Se você definir a propriedade como `true`, o Loader verificará o servidor de origem para obter um arquivo de política (consulte“Controles de site (arquivos de política)” na página 1089). Isso é necessário apenas para o conteúdo originado de domínios diferentes dos domínios do arquivo SWF que contém o objeto Loader. Se o servidor conceder permissão ao domínio de Loader, o ActionScript dos arquivos SWF do domínio de Loader poderá acessar os dados na imagem carregada; em outras palavras, você pode usar o comando `BitmapData.draw()` para acessar os dados na imagem carregada.

  Observe que um arquivo SWF de outros domínios que não são os do objeto Loader pode chamar `Security.allowDomain()` para permitir um domínio específico.

- `securityDomain`: Só use essa propriedade ao carregar um arquivo SWF (não uma imagem). Especifique-a para um arquivo SWF de um domínio diferente daquele do arquivo que contém o objeto Loader. Quando essa opção é especificada, o Flash Player verifica a existência de um arquivo de política e, se existir algum, os arquivos SWF dos domínios permitidos no arquivo de política poderão cruzar o script do conteúdo SWF carregado. Você pode especificar `flash.system.SecurityDomain.currentDomain` como este parâmetro.

- `applicationDomain`: Use essa propriedade apenas ao carregar um arquivo SWF gravado no ActionScript 3.0 (não em uma imagem ou arquivo SWF gravado no ActionScript 1.0 ou no ActionScript 2.0). Ao carregar o arquivo, você pode especificar que o arquivo seja incluído no mesmo domínio de aplicativo do objeto Loader, definindo o parâmetro `applicationDomain` como `flash.system.ApplicationDomain.currentDomain`. Colocando o arquivo SWF carregado no mesmo domínio de aplicativo, é possível acessar suas classes diretamente. Isso pode ser útil se estiver carregando um arquivo SWF que contém mídia incorporada, que pode ser acessada por meio dos nomes de classe associados. Para obter mais informações, consulte "Trabalhar com domínios de aplicativo" na página 149.

Veja um exemplo de busca de um arquivo de política ao carregar um bitmap de outro domínio:

```
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);
```

Veja um exemplo de busca de um arquivo de política ao carregar um SWF de outro domínio para colocar o arquivo na mesma caixa de proteção do objeto Loader. Além disso, o código adiciona as classes do arquivo SWF carregado ao mesmo domínio de aplicativo do objeto Loader:

```
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);
```

Para obter mais informações, consulte a classe LoaderContext em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Carregamento de arquivos SWF no AIR for iOS

Adobe AIR 3.6 e posterior, somente iOS

Nos dispositivos iOS, há restrições no código de carregamento e compilação no runtime. Devido a essas restrições, existem certas diferenças necessárias na tarefa de carregar arquivos SWF externos em seu aplicativo:

- Todos os arquivos SWF que contenham o código ActionScript devem ser incluídos no pacote do aplicativo. Nenhum SWF com código pode ser carregado de uma fonte externa, como uma rede. Como parte do empacotamento do arquivo, todo código ActionScript em todos os arquivos SWF no pacote do aplicativo é compilado no código nativo para dispositivos iOS.
- Não é possível carregar, descarregar e, em seguida, recarregar um arquivo SWF. Caso tente fazer isso, ocorrerá um erro.
- O comportamento de carregar na memória e descarregar é o mesmo de plataformas de desktop. Se você carregar um arquivo SWF e, em seguida, descarregá-lo, todos os recursos visuais contidos no SWF serão descarregados da memória. Contudo, todas as referências de classe para uma classe ActionScript no SWF carregado permanecem na memória e podem ser acessadas no código ActionScript.
- Todos os arquivos SWF carregados devem ser carregados no mesmo domínio de aplicativo do arquivo SWF principal. Como esse não é o comportamento padrão, para cada SWF que você carregar, deverá criar um objeto LoaderContext especificando o principal domínio de aplicativo e enviar esse objeto LoaderContext para a chamada de método Loader.load(). Se você tentar carregar um SWF em um domínio de aplicativo que não seja o do SWF principal, ocorrerá um erro. Isso ocorre mesmo se o SWF carregado contiver apenas recursos visuais e nenhum código ActionScript.

O exemplo a seguir apresenta o código a ser utilizado para carregar um SWF de um pacote de aplicativo para o domínio de aplicativo do SWF principal.

```
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);
```

Um arquivo SWF que possua apenas recursos visuais e nenhum código pode ser carregado do pacote do aplicativo ou de uma rede. Em ambos os casos, o arquivo SWF ainda assim deverá ser carregado no domínio do aplicativo principal.

No caso de versões do AIR anteriores ao AIR 3.6, todo o código é removido de SWFs diferentes do SWF do aplicativo principal durante ao processo de compilação. Os arquivos SWF que contenham somente recursos visuais podem ser incluídos no pacote do aplicativo e carregados no runtime, mas sem código. Se você tentar carregar um SWF que possua um código ActionScript, ocorrerá um erro. O erro faz com que surja uma caixa de diálogo de erro "ActionScript descompilado" no aplicativo.

Consulte também

Empacotamento e carregamento de vários SWFs em aplicativos do AIR no iOS

Uso das classes ProLoader e ProLoaderInfo

Flash Player 9 e superior, Adobe AIR 1.0 e superior, e requer o Flash Professional CS5.5

Para ajudar no precarregamento da biblioteca de compartilhamento remoto (RSL - remote shared library), o Flash Professional CS5.5 introduz as classes fl.display.ProLoader and fl.display.ProLoaderInfo. Essas classes refletem as classes flash.display.Loader and flash.display.LoaderInfo, mas proporcionam uma experiência de carregamento mais consistente.

Em particular, a classe ProLoader ajuda-o a carregar arquivos SWF que utilizam Estrutura de Layout de Texto (TLF - Text Layout Framework) com precarregamento da RSL preloading. No tempo de execução, os arquivos SWF que carregam outros arquivos SWF ou arquivos SWZ, tais como TLF, requirem um arquivo de invólucro SWF somente para uso interno. A camada extra de complexidade imposta pelo arquivo de invólucro SWF pode causar um comportamento indesejável. A classe ProLoader resolve essa complexidade no carregamento desses arquivos como se eles fossem arquivos SWF comuns. A solução usada pela classe ProLoader é transparente para o usuário e não requer nenhum processamento especial no ActionScript. Além disso, a classe ProLoader carrega o conteúdo SWF comum corretamente.

No Flash Professional CS5.5 e posteriores, é possível substituir com segurança todas as utilizações da classe Loader pela classe ProLoader. Em seguida, você pode exportar o aplicativo Flash Player 10.2 ou posgterior para que a classe ProLoader acesse a funcionalidade do ActionScript necessária. Também é possível utilizar a classe ProLoader ao definir como destino versões anteriores do Flash Player que suportam o ActionScript 3.0. Contudo, as vantagens completas dos recursos da classe ProLoader só podem ser obtidas com o Flash Player 10.2 ou posterior. Use sempre ProLoader ao usar TLF no Flash Professional CS5.5 ou posterior. A classe ProLoader não é necessária em ambientes estranhos ao Flash Professional.

Importante: *Para arquivos SWF publicados no Flash Professional CS5.5 e posterior, você pode sempre usar as classes fl.display.ProLoader e fl.display.ProLoaderInfo em vez de flash.display.Loader e flash.display.LoaderInfo.*

Problemas tratados pela classe Proloader

A classe ProLoader trata de problemas que a classe legada Loader não se destinava a tratar. Esses problemas decorrem do precarregamento da RSL das bibliotecas TLF. Especificamente, elas se aplicam aos arquivos SWF que usam um objeto Loader para carregar outros arquivos SWF. Os problemas tratados incluem os seguintes:

- **Os scripts entre o arquivo de carregamento e o arquivo carregado não se comportam como esperado.** A classe ProLoader define automaticamente o arquivo SWF de carregamento como pai do arquivo SWF carregado. Desse modo, as comunicações do arquivo SWF de carregamento vão diretamente para o arquivo SWF carregado.
- **O aplicativo SWF deve gerenciar ativamente o processo de carregamento.** Isso requer a implantação de eventos adicionais como `added`, `removed`, `addedToStage` e `removedFromStage`. Se o seu aplicativo for desenvolvido para o Flash Player 10.2 ou superior, a classe ProLoader eliminará a necessidade desse trabalho extra.

Atualizando o código para utilizar ProLoader em vez de Loader

Uma vez que a classe ProLoader reflete a classe, você pode alternar facilmente as duas classes no seu código. O exemplo a seguir mostra como atualizar um código existente para utilizar a nova classe:

```
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!');
}
```

Esse código pode ser atualizado para usar ProLoader como segue:

```
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!');
}
```

Exemplo de objeto de exibição: SpriteArranger

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo de amostra SpriteArranger é criado com base no aplicativo de exemplo Geometric Shapes descrito separadamente no *Aprendizado do ActionScript 3.0*.

O aplicativo de amostra SpriteArranger ilustra diversos conceitos de manipulação de objetos de exibição:

- Extensão de classes de objeto de exibição
- Adição de objetos à lista de exibição
- Disposição em camadas de objetos de exibição e trabalho com contêineres de objeto de exibição

- Resposta a eventos de objeto de exibição
- Uso de propriedades e métodos de objetos de exibição

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo SpriteArranger estão localizados na pasta Exemplos/SpriteArranger. O aplicativo consiste nos seguintes arquivos:

Arquivo	**Descrição**
SpriteArranger.mxml ou SpriteArranger.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/SpriteArranger/CircleSprite.as	Uma classe que define um tipo de objeto Sprite que processa um círculo na tela.
com/example/programmingas3/SpriteArranger/DrawingCanvas.as	Uma classe que define a tela de desenho, que é um contêiner de objeto de exibição que contém objetos GeometricSprite.
com/example/programmingas3/SpriteArranger/SquareSprite.as	Uma classe que define um tipo de objeto Sprite que processa um quadrado na tela.
com/example/programmingas3/SpriteArranger/TriangleSprite.as	Uma classe que define um tipo de objeto Sprite que processa um triângulo na tela.
com/example/programmingas3/SpriteArranger/GeometricSprite.as	Uma classe que estende o objeto Sprite, usado para definir uma forma na tela. CircleSprite, SquareSprite e TriangleSprite estendem essa classe.
com/example/programmingas3/geometricshapes/IGeometricShape.as	A interface básica que define os métodos a serem implementados por todas as classes de forma geométrica.
com/example/programmingas3/geometricshapes/IPolygon.as	Uma interface que define os métodos a serem implementados pelas classes de forma geométrica que têm vários lados.
com/example/programmingas3/geometricshapes/RegularPolygon.as	Um tipo de forma geométrica que tem lados com o mesmo comprimento posicionados simetricamente ao redor do centro da forma.
com/example/programmingas3/geometricshapes/Circle.as	Um tipo de forma geométrica que define um círculo.
com/example/programmingas3/geometricshapes/EquilateralTriangle.as	Uma subclasse de RegularPolygon que define um triângulo com todos os lados com o mesmo comprimento.
com/example/programmingas3/geometricshapes/Square.as	Uma subclasse de RegularPolygon que define um retângulo com os quatro lados com o mesmo comprimento.
com/example/programmingas3/geometricshapes/GeometricShapeFactory.as	Uma classe que contém um “método de fábrica” para criar formas com tamanho e tipo especificados.

Definição das classes SpriteArranger

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo SpriteArranger permite que o usuário adicione vários objetos de exibição à "tela de desenho" exibida.

A classe DrawingCanvas define uma área de desenho, um tipo de contêiner de objeto de exibição ao qual o usuário pode adicionar formas na tela. Essas formas na tela são ocorrências de uma das subclasses de GeometricSprite.

A classe DrawingCanvas

No Flex, todos os objetos de exibição filho adicionados a um objeto Container devem ser de uma classe originada da classe mx.core.UIComponent. Esse aplicativo adiciona uma ocorrência da classe DrawingCanvas como um filho de um objeto mx.containers.VBox, conforme definido no código MXML no arquivo SpriteArranger.mxml. Essa herança é definida na declaração da classe DrawingCanvas, do seguinte modo:

```
public class DrawingCanvas extends UIComponent
```

A classe UIComponent é herdada das classes DisplayObject, DisplayObjectContainer e Sprite e o código na classe DrawingCanvas usa métodos e propriedades dessas classes.

A classe DrawingCanvas estende a classe Sprite e sua herança é definida na declaração da classe DrawingCanvas, do seguinte modo:

```
public class DrawingCanvas extends Sprite
```

A classe Sprite é uma subclasse de DisplayObjectContainer e de DisplayObject, e a classe DrawingCanvas usa métodos e propriedades dessas classes.

O método do construtor `DrawingCanvas()` configura um objeto de Rectangle, `bounds`, que é a propriedade usada posteriormente no desenho do contorno da tela de desenho. Em seguida, o método `initCanvas()` é chamado do seguinte modo:

```
this.bounds = new Rectangle(0, 0, w, h);
initCanvas(fillColor, lineColor);
```

Como mostra o exemplo a seguir, o método `initCanvas()` define várias propriedades do objeto DrawingCanvas, que foram transmitidas como argumentos para a função do construtor:

```
this.lineColor = lineColor;
this.fillColor = fillColor;
this.width = 500;
this.height = 200;
```

O método `initCanvas()` chama o método `drawBounds()`, que desenha a tela de desenho usando a propriedade `graphics` da classe DrawingCanvas. A propriedade `graphics` é herdada da classe Shape.

```
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();
```

Os métodos adicionais a seguir da classe DrawingCanvas são invocados com base nas interações do usuário com o aplicativo:

- Os métodos `addShape()` e `describeChildren()`, que são descritos em "Adição de objetos de exibição à tela de desenho" na página 210
- Os métodos `moveToBack()`, `moveDown()`, `moveToFront()` e `moveUp()`, que são descritos em "Reorganização da disposição em camadas do objeto de exibição" na página 213
- O método `onMouseUp()`, que é descrito em "Clique e arrasto de objetos de exibição" na página 212

A classe GeometricSprite e suas subclasses

Cada objeto de exibição que pode ser adicionado à tela de desenho pelo usuário é uma ocorrência de uma das seguintes subclasses de GeometricSprite:

- CircleSprite
- SquareSprite
- TriangleSprite

A classe GeometricSprite estende a classe flash.display.Sprite:

```
public class GeometricSprite extends Sprite
```

A classe GeometricSprite inclui diversas propriedades comuns a todos os objetos GeometricSprite. Essas propriedades são definidas na função do construtor com base nos parâmetros transmitidos para a função. Por exemplo:

```
this.size = size;
this.lineColor = lColor;
this.fillColor = fColor;
```

A propriedade `geometricShape` da classe GeometricSprite define uma interface IGeometricShape, que define as propriedades matemáticas, mas não as visuais, da forma. As classes que implementam a interface IGeometricShape são definidas no aplicativo de amostra GeometricShapes descrito no *Aprendizado do ActionScript 3.0*.

A classe GeometricSprite define o método `drawShape()`, que é refinado ainda mais nas definições de substituição em cada subclasse de GeometricSprite. Para obter mais informações, consulte a seção "Adição de objetos de exibição à tela de desenho", apresentada a seguir.

A classe GeometricSprite também fornece os seguintes métodos:

- Os métodos `onMouseDown()` e `onMouseUp()`, que são descritos em "Clique e arrasto de objetos de exibição" na página 212
- Os métodos `showSelected()` e `hideSelected()`, que são descritos em "Clique e arrasto de objetos de exibição" na página 212

Adição de objetos de exibição à tela de desenho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o usuário clica no botão Adicionar forma, o aplicativo chama o método `addShape()` da classe DrawingCanvas. Esse método percorre um novo GeometricSprite chamando a função do construtor adequada de uma das subclasses de GeometricSprite, como mostra o exemplo a seguir:

```
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);
}
```

Cada método do construtor chama o método `drawShape()`, que usa a propriedade `graphics` da classe (herdada da classe Sprite) para desenhar o gráfico vetorial adequado. Por exemplo, o método `drawShape()` da classe CircleSprite inclui o seguinte código:

```
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);
```

Da segunda à última linha da função `addShape()`, é definida a propriedade `alpha` do objeto de exibição (herdada da classe DisplayObject) para que cada objeto de exibição adicionado à tela de desenho seja ligeiramente transparente, deixando o usuário ver os objetos que estão por trás.

A linha final do método `addChild()` adiciona o novo objeto de exibição à lista de filhos da ocorrência da classe DrawingCanvas, que já está na lista de exibição. Desse modo, o novo objeto de exibição aparece no palco.

A interface do aplicativo inclui dois campos de texto, `selectedSpriteTxt` e `outputTxt`. As propriedades de texto desses campos são atualizados com informações sobre os objetos GeometricSprite que foram adicionados à tela de desenho ou selecionados pelo usuário. A classe GeometricSprite manipula essa tarefa de registro de informações substituindo o método `toString()` da seguinte maneira:

```
public override function toString():String
{
    return this.shapeType + " of size " + this.size + " at " + this.x + ", " + this.y;
}
```

A propriedade `shapeType` é definida como o valor adequado no método do construtor de cada subclasse de GeometricSprite. Por exemplo, o método `toString()` poderia retornar o seguinte valor para uma ocorrência de CircleSprite adicionada recentemente à ocorrência de DrawingCanvas:

```
Circle of size 50 at 0, 0
```

O método `describeChildren()` da classe DrawingCanvas percorre a lista de filhos da tela de desenho, usando a propriedade `numChildren` (herdada da classe DisplayObjectContainer), para definir o limite do loop `for`. É gerada uma string que lista cada filho, do seguinte modo:

```
var desc:String = "";
var child:DisplayObject;
for (var i:int=0; i < this.numChildren; i++)
{
child = this.getChildAt(i);
desc += i + ": " + child + '\n';
}
```

A string resultante é usada para definir a propriedade `text` do campo de texto `outputTxt`.

Clique e arrasto de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o usuário clica em uma ocorrência de GeometricSprite, o aplicativo chama o manipulador de eventos `onMouseDown()`. Conforme mostrado a seguir, esse manipulador de eventos é definido para ouvir eventos de mouse na função do construtor da classe GeometricSprite:

```
this.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
```

O método `onMouseDown()` chama o método `showSelected()` do objeto GeometricSprite. Se for a primeira vez que esse método foi chamado para o objeto, o método criará um novo objeto Shape chamado `selectionIndicator` e usará a propriedade `graphics` do objeto Shape para desenhar um retângulo de realce vermelho, do seguinte modo:

```
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);
```

Se não for a primeira vez que o método `onMouseDown()` é chamado, o método simplesmente definirá a propriedade `visible` da forma `selectionIndicator` (herdada da classe DisplayObject), do seguinte modo:

```
this.selectionIndicator.visible = true;
```

O método `hideSelected()` oculta a forma `selectionIndicator` do objeto selecionado anteriormente definindo a propriedade `visible` como `false`.

O manipulador de eventos `onMouseDown()` também chama o método `startDrag()` (herdado da classe Sprite), que inclui o seguinte código:

```
var boundsRect:Rectangle = this.parent.getRect(this.parent);
boundsRect.width -= this.size;
boundsRect.height -= this.size;
this.startDrag(false, boundsRect);
```

Isso permite que o usuário arraste o objeto selecionado em torno da tela de desenho, dentro dos limites definidos pelo retângulo `boundsRect`.

Quando o usuário solta o botão do mouse, o evento `mouseUp` é enviado. O método do construtor de DrawingCanvas configura o seguinte ouvinte de eventos:

```
this.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
```

Esse ouvinte de eventos é definido para o objeto DrawingCanvas, não para objetos GeometricSprite individuais. Isso ocorre porque quando o objeto GeometricSprite é arrastado, pode terminar atrás de outro objeto de exibição (outro objeto GeometricSprite) assim que o mouse é solto. O objeto de exibição em primeiro plano receberia o evento de mouse, mas o objeto de exibição que está sendo arrastado pelo usuário não. A adição do ouvinte ao objeto DrawingCanvas garante que o evento seja sempre manipulado.

O método `onMouseUp()` chama o método `onMouseUp()` do objeto GeometricSprite, que, por sua vez, chama o método `stopDrag()` do objeto GeometricSprite.

Reorganização da disposição em camadas do objeto de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A interface de usuário do aplicativo inclui os botões Mover para trás, Mover para baixo, Mover para cima e Mover para frente. Quando o usuário clica em um desses botões, o aplicativo chama o método correspondente da classe DrawingCanvas: `moveToBack()`, `moveDown()`, `moveUp()` ou `moveToFront()`. Por exemplo, o método `moveToBack()` inclui o seguinte código:

```
public function moveToBack(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, 0);
    }
}
```

O método `setChildIndex()` (herdado da classe DisplayObjectContainer) é usado para colocar o objeto de exibição na posição de índice 0 na lista de filhos da ocorrência de DrawingCanvas (`this`).

O método `moveDown()` funciona de modo similar, mas diminui a posição de índice do objeto de exibição em incrementos de 1 na lista de filhos da ocorrência de DrawingCanvas:

```
public function moveDown(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, index - 1);
    }
}
```

Os métodos `moveUp()` e `moveToFront()` funcionam de modo similar aos métodos `moveToBack()` e `moveDown()`.

Capítulo 11: Trabalho com geometria

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O pacote flash.geom contém classes que definem objetos geométricos como pontos, retângulos e matrizes de transformação. Essas classes são usadas para definir as propriedades dos objetos utilizados em outras classes.

Mais tópicos da Ajuda

Pacote flash.geom

Noções básicas de geometria

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O pacote flash.geom contém classes que definem objetos geométricos como pontos, retângulos e matrizes de transformação. Essas classes necessariamente não fornecem uma funcionalidade em si, mas são usadas para definir as propriedades dos objetos utilizados em outras classes.

Todas as classes de geometria giram em torno da noção de que os locais na tela são representados como um plano bidimensional. A tela é tratada como um gráfico plano com um eixo horizontal (x) e um vertical (y). Qualquer local (ou *ponto*) da tela pode ser representado como um par de valores x e y — as *coordenadas* desse local.

Cada objeto de exibição, inclusive o palco, tem seu próprio *espaço de coordenadas*. O espaço de coordenadas é o próprio gráfico de um objeto para traçar os locais dos objetos de exibição descendentes, desenhos, etc. A *origin* está na coordenada 0, 0 (onde os eixos x e y se encontram) e está localizada no canto superior esquerdo do objeto de exibição. Embora esse local de origem seja sempre válido para o palco, não é necessariamente válido para nenhum outro objeto de exibição. Os valores do eixo x aumentam em direção à direita e diminuem em direção à esquerda. Para os locais à esquerda da origem, a coordenada x é negativa. Entretanto, ao contrário do sistema tradicional de coordenadas, os valores de coordenadas do tempo de execução do Flash no eixo y aumentam quando descem na tela e diminuem quando sobem na tela. Os valores acima da origem têm um valor negativo de coordenada y). Como o canto superior esquerdo do palco é a origem do espaço de coordenadas, a maioria dos objetos no palco tem uma coordenada x maior que 0 e menor que a largura do palco. E o mesmo objeto tem uma coordenada y maior que 0 e menor que a altura do palco.

Você pode usar ocorrências da classe Point para representar pontos individuais em um espaço de coordenadas. É possível criar uma ocorrência de Rectangle para representar uma região retangular em um espaço de coordenadas. Para usuários avançados, você pode usar uma ocorrência de Matrix para aplicar várias transformações ou transformações complexas em um objeto de exibição. Muitas transformações simples, como rotação, posição e alterações de escala, podem ser aplicadas diretamente a um objeto de exibição usando as propriedades desse objeto. Para obter informações sobre como aplicar transformações usando propriedades de objeto de exibição, consulte "Manipulação de objetos de exibição" na página 175.

Conceitos e termos importantes

A lista de referências a seguir contém termos geométricos importantes:

Coordenadas cartesianas Coordenadas comumente escritas como um par de números (como 5, 12 ou 17, -23). Os dois números são as coordenadas x e y, respectivamente.

Espaço de coordenadas O gráfico de coordenadas contido em um objeto de exibição, no qual seus elementos-filho são posicionados.

Origem O ponto em um espaço de coordenadas onde o eixo x encontra o eixo y. Esse ponto tem a coordenada 0,0.

Ponto Um local isolado em um espaço de coordenadas. No sistema de coordenadas bidimensional usado no ActionScript, o local ao longo do eixo x e do eixo y (as coordenadas do ponto) define o ponto.

Ponto de registro Em um objeto de exibição, a origem (coordenada 0, 0) de seu espaço coordenado.

Dimensionamento O tamanho de um objeto em relação ao seu tamanho original. Quando usado como um verbo, dimensionar um objeto significa alterar seu tamanho, aumentando ou diminuindo o objeto.

Traduzir Alterar as coordenadas de um ponto de um espaço de coordenadas para outro.

Transformação Ajuste de uma característica visual de um gráfico como a rotação do objeto, a alteração da escala, a inclinação ou distorção da forma ou a alteração da cor.

Eixo X O eixo horizontal no sistema de coordenadas bidimensional usado no ActionScript.

Eixo Y O eixo vertical no sistema de coordenadas bidimensional usado no ActionScript.

Uso de objetos Point

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O objeto Point define um par de coordenadas Cartesianas. Ele representa um local em um sistema de coordenadas bidimensional, em que *x* representa o eixo horizontal e *y*, o vertical.

Para definir um objeto Point, defina as propriedades de `x` e `y` da seguinte forma:

```
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;
```

Cálculo da distância entre dois pontos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar o método `distance()` da classe Point para calcular a distância entre dois pontos em um espaço de coordenadas. Por exemplo, o código a seguir calcula a distância entre os pontos de registro de dois objetos de exibição, `circle1` e `circle2`, no mesmo contêiner de objetos de exibição:

```
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);
```

Transposição de espaços de coordenadas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se dois objetos de exibição estiverem em contêineres de objetos de exibição diferentes, talvez estejam em espaços de coordenadas diferentes. Você pode usar o método `localToGlobal()` da classe DisplayObject para transpor as coordenadas para o mesmo espaço de coordenadas (global) que o Palco. Por exemplo, o código a seguir calcula a distância entre os pontos de registro de dois objetos de exibição, `circle1` e `circle2`, em contêineres de objetos de exibição diferentes:

```
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);
```

Da mesma forma, para calcular a distância do ponto de registro de um objeto de exibição chamado `target` a partir de um ponto específico no palco, use o método `localToGlobal()` da classe DisplayObject:

```
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);
```

Mover um objeto de exibição usando um ângulo e uma distância especificados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar o método `polar()` da classe Point para mover um objeto de exibição a uma distância específica em um ângulo específico. Por exemplo, o código a seguir move o objeto `myDisplayObject` 100 pixels em 60°:

```
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;
```

Outros usos da classe Point

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar os objetos Point com os seguintes métodos e propriedades:

Classe	Métodos ou propriedades	Descrição
DisplayObjectContainer	`areInaccessibleObjectsUnderPoint()` `getObjectsUnderPoint()`	Usada para retornar uma lista de objetos sob um ponto em um contêiner de objetos de exibição.
BitmapData	`hitTest()`	Usada para definir o pixel no objeto BitmapData bem como o ponto cuja ocorrência você está verificando.

BitmapData	`applyFilter()`
	`copyChannel()`
	`merge()`
	`paletteMap()`
	`pixelDissolve()`
	`threshold()`
		Usada para definir as posições de retângulos que definem as operações.

Matriz	`deltaTransformPoint()` `transformPoint()`	Usada para definir os pontos para os quais você deseja aplicar uma transformação.

Retângulo	`bottomRight`
	`size`
	`topLeft`
		Usada para definir essas propriedades.

Uso de objetos Rectangle

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O objeto Rectangle define uma área retangular. Um objeto Rectangle tem uma posição, definida pelas coordenadas *x* e *y* de seu canto superior esquerdo, uma propriedade `width` e uma propriedade `height` . Você pode definir essas propriedades para um novo objeto Rectangle chamando a função de construtor `Rectangle()` da seguinte forma:

```
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);
```

Redimensionamento e reposicionamento de objetos Rectangle

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Há várias formas de redimensionar e reposicionar objetos Rectangle.

Você pode reposicionar o objeto Rectangle alterando suas propriedades `x` e `y`. Essa alteração não afeta a largura ou a altura do objeto Rectangle.

```
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)
```

Como mostra o código a seguir, quando você altera a propriedade `left` ou `top` de um objeto Rectangle, o retângulo é reposicionado. As propriedades `x` e `y` do retângulo correspondem às propriedades `left` e `top`, respectivamente. Entretanto, a posição do canto inferior esquerdo do objeto Rectangle não é alterada, por isso ele é redimensionado.

```
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)
```

Da mesma forma, como mostra o exemplo, se você alterar a propriedade `bottom` ou `right` de um objeto Rectangle, a posição de seu canto superior esquerdo não é alterada. O retângulo é realizado corretamente:

```
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)
```

Você também pode reposicionar um objeto Rectangle usando o método `offset()` da seguinte maneira:

```
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)
```

O método `offsetPt()` funciona da mesma forma, exceto que assume um objeto Point como parâmetro, em vez dos valores de deslocamento *x* e *y*.

Você também pode redimensionar um objeto Rectangle usando o método `inflate()`, que inclui dois parâmetros, `dx` e `dy`. O parâmetro `dx` representa o número de pixels que os lados esquerdo e direito do retângulo se movem em relação ao centro. O parâmetro `dy` representa o número de pixels que os lados superior e inferior do retângulo se movem em relação ao centro.

```
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)
```

O método `inflatePt()` funciona da mesma forma, exceto que assume um objeto Point como parâmetro, em vez dos valores de deslocamento `dx` e `dy`.

Localização de uniões e interseções de objetos Rectangle

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você usa o método `union()` para localizar a região retangular formada pelos limites de dois retângulos:

```
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)
```

Você usa o método `intersection()` para localizar a região retangular formada pela região sobreposta de dois retângulos:

```
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)
```

O método `intersects()` é usado para descobrir se há interseção de dois retângulos. O método `intersects()` também pode ser usado para descobrir se um objeto de exibição está em uma determinada região do Palco. No código de exemplo a seguir, suponha que o espaço de coordenadas do contêiner de objetos de exibição que inclui o objeto `circle` seja o mesmo do palco. O exemplo mostra como usar o método `intersects()` para determinar se há interseção de um objeto de exibição, `circle`, com regiões especificadas do Palco, definidas pelos objetos Rectangle `target1` e `target2`:

```
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
```

Da mesma forma, o método `intersects()` pode ser usado para descobrir se os retângulos delimitadores de dois objetos de exibição se sobrepõem. Use o método `getRect()` da classe DisplayObject para incluir qualquer espaço adicional que os traçados de um objeto de exibição adicionam a uma região delimitadora.

Outros usos de objetos Rectangle

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os objetos Rectangle são usados nos seguintes métodos e propriedades:

Classe	Métodos ou propriedades	Descrição
BitmapData	`applyFilter()`, `colorTransform()`, `copyChannel()`, `copyPixels()`, `draw()`, `drawWithQuality()`, `encode()`, `fillRect()`, `generateFilterRect()`, `getColorBoundsRect()`, `getPixels()`, `merge()`, `paletteMap()`, `pixelDissolve()`, `setPixels()` e `threshold()`	Usada como o tipo para alguns parâmetros a fim de definir uma região do objeto BitmapData.
DisplayObject	`getBounds()`, `getRect()`, `scrollRect`, `scale9Grid`	Usada como o tipo de dados para a propriedade ou o tipo de dados retornado.
PrintJob	`addPage()`	Usada para definir o parâmetro `printArea`.
Sprite	`startDrag()`	Usada para definir o parâmetro `bounds`.
TextField	`getCharBoundaries()`	Usada como um tipo de valor de retorno.
Transform	`pixelBounds`	Usada como o tipo de dados.

Uso de objetos Matrix

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Matrix representa uma matriz de transformação que determina como o map aponta de um espaço de coordenada para outro. É possível realizar várias transformações gráficas em um objeto de exibição definindo as propriedades de um objeto Matrix, aplicando esse objeto Matrix à propriedade `matrix` de um objeto Transform, e depois aplicando esse objeto Transform como a propriedade `transform` do objeto de exibição. Essas funções de transformação incluem conversão (reposicionamento de *x* e *y*), rotação, dimensionamento e inclinação.

Embora seja possível definir uma matriz ajustando diretamente as propriedades (`a`, `b`, `c`, `d`, `tx`, `ty`) de um objeto Matrix, é mais fácil usar o método `createBox()`. Esse método inclui parâmetros que permitem definir diretamente os efeitos de dimensionamento, rotação e transposição da matriz resultante. Por exemplo, o código a seguir cria um objeto Matrix que dimensiona um objeto horizontalmente em 2,0, verticalmente em 3,0, gira-o em 45°, movendo-o (transpondo) 10 pixels para a direita e 20 pixels para baixo:

```
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);
```

Também é possível ajustar os efeitos de dimensionamento, rotação e transposição de um objeto Matrix usando os métodos `scale()`, `rotate()` e `translate()` . Observe que esses métodos combinam com os valores do objeto Matrix existente. Por exemplo, o código a seguir define um objeto Matrix que dimensiona um objeto por um fator de 4 e gira-o 60°, desde que os métodos `scale()` e `rotate()` sejam chamados duas vezes:

```
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;
```

Para aplicar uma transformação de inclinação em um objeto Matrix, ajuste sua propriedade `b` ou `c`. O ajuste da propriedade `b` inclina a matriz verticalmente e o ajuste da propriedade`c` inclina a matriz horizontalmente. O código a seguir inclina o objeto Matrix `myMatrix` verticalmente por um fator de 2:

```
var skewMatrix:Matrix = new Matrix();
skewMatrix.b = Math.tan(2);
myMatrix.concat(skewMatrix);
```

Você pode aplicar uma transformação de Matrix à propriedade `transform` de um objeto de exibição. Por exemplo, o código a seguir aplica uma transformação de matriz em um objeto de exibição chamado `myDisplayObject`:

```
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;
```

A primeira linha define um objeto Matrix como a matriz de transformação existente usada pelo objeto de exibição `myDisplayObject` (a propriedade `matrix` da propriedade `transformation` do objeto de exibição `myDisplayObject`). Dessa forma, os métodos da classe Matrix que você chama terão um efeito cumulativo na posição, dimensão e rotação existentes do objeto de exibição.

Nota: *A classe ColorTransform também está incluída no pacote flash.geometry. Essa classe é usada para definir a propriedade `colorTransform` de um objeto Transform. Como não se aplica a nenhuma transformação geométrica, ela não será discutida em detalhes aqui. Para obter mais informações, consulte a classe ColorTransform em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.*

Exemplo de geometria: aplicação de uma transformação de matriz em um objeto de exibição

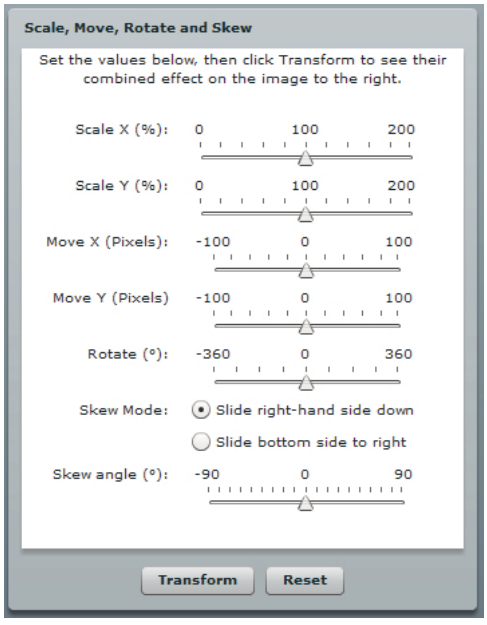
Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo de exemplo DisplayObjectTransformer mostra vários recursos de uso da classe Matrix para transformar um objeto de exibição, incluindo o seguinte:

- Girar o objeto de exibição
- Dimensionar o objeto de exibição

- Transpor (reposicionar) o objeto de exibição
- Inclinar o objeto de exibição

O aplicativo fornece uma interface para ajustar os parâmetros de transformação da matriz, como a seguir:



Quando o usuário clica no botão Transform, o aplicativo aplica a transformação apropriada.

O objeto de exibição original e o objeto de exibição com um giro de - 45° e um dimensionamento de 50%

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo DisplayObjectTransformer podem ser encontrados na pasta Samples/DisplayObjectTransformer. O aplicativo consiste nos seguintes arquivos:

Arquivo	**Descrição**
DisplayObjectTransformer.mxml ou DisplayObjectTransformer.fla	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML)
com/example/programmingas3/geometry/MatrixTransformer.as	Uma classe que contém métodos para aplicar transformações de matriz.
img/	Um diretório contendo arquivos de imagem de exemplo usados pelo aplicativo.

Definição da classe MatrixTransformer

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe MatrixTransformer inclui métodos estáticos que aplicam transformações geométricas de objetos Matrix.

O método transform()

O método `transform()` inclui parâmetros para cada um dos itens a seguir:

- `sourceMatrix` — a matriz de entrada, que o método transforma
- `xScale` e `yScale` — o fator de dimensionamento *x* e *y*
- `dx` e `dy` — os valores de transposição *x* e *y*, em pixels
- `rotation` — o valor de rotação, em graus
- `skew` — o fator de inclinação, em porcentagem
- `skewType` — a direção da inclinação `"right"` ou `"left"`

O valor de retorno é a matriz resultante.

O método `transform()` chama os seguintes métodos estáticos da classe:

- `skew()`
- `scale()`
- `translate()`
- `rotate()`

Cada um retorna a matriz de origem com a transformação aplicada.

O método skew()

O método `skew()` inclina a matriz, ajustando as propriedades `b` e `c` da matriz. Um parâmetro opcional, `unit`, determina as unidades usadas para definir o ângulo de inclinação e, se necessário, o método converte o valor `angle` em radianos:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
```

Um objeto Matrix `skewMatrix` é criado e ajustado para aplicar a transformação de inclinação. Inicialmente, é a matriz de identidade, como a seguir:

```
var skewMatrix:Matrix = new Matrix();
```

O parâmetro `skewSide` determina o lado para o qual a inclinação é aplicada. Se for definida como `"right"`, o seguinte código define a propriedade `b` da matriz:

```
skewMatrix.b = Math.tan(angle);
```

Caso contrário, o lado inferior é inclinado ajustando a propriedade `c` de Matrix, como a seguir:

```
skewMatrix.c = Math.tan(angle);
```

A inclinação resultante é aplicada à matriz existente, concatenando as duas matrizes, como mostra o seguinte exemplo:

```
sourceMatrix.concat(skewMatrix);
return sourceMatrix;
```

O método scale()

O exemplo a seguir mostra que o método `scale()` primeiro ajusta o fator de dimensionamento, caso seja fornecido como uma porcentagem e, em seguida, usa o método `scale()` do objeto da matriz:

```
if (percent)
{
    xScale = xScale / 100;
    yScale = yScale / 100;
}
sourceMatrix.scale(xScale, yScale);
return sourceMatrix;
```

O método translate()

O método `translate()` apenas aplica os fatores de transposição `dx` e `dy` chamando o método `translate()` do objeto da matriz, como a seguir:

```
sourceMatrix.translate(dx, dy);
return sourceMatrix;
```

O método rotate()

O método `rotate()` converte o fator de rotação de entrada em radianos (se fornecido em graus ou gradientes) e chama o método `rotate()` do objeto da matriz:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
sourceMatrix.rotate(angle);
return sourceMatrix;
```

Chamada do método MatrixTransformer.transform() no aplicativo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo contém uma interface do usuário para obter os parâmetros de transformação do usuário. Então, ele transmite esses valores, junto com a propriedade `matrix` da propriedade `transform` do objeto de exibição, ao método `Matrix.transform()`, da seguinte maneira:

```
tempMatrix = MatrixTransformer.transform(tempMatrix,
    xScaleSlider.value,
    yScaleSlider.value,
    dxSlider.value,
    dySlider.value,
    rotationSlider.value,
    skewSlider.value,
    skewSide );
```

O aplicativo aplica o valor de retorno à propriedade `matrix` da propriedade `transform` do objeto de exibição, acionando a transformação:

```
img.content.transform.matrix = tempMatrix;
```

Capítulo 12: Uso da API de desenho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Embora as imagens e a arte final importadas sejam importantes, a funcionalidade conhecida como API de desenho, que permite desenhar linhas e formas no ActionScript, dá a você a liberdade de iniciar um aplicativo com o equivalente computacional de uma tela de desenho em branco, na qual é possível criar as imagens desejadas. A habilidade de criar seus próprios gráficos abre muitas possibilidades para seus aplicativos. Com as técnicas tratadas aqui, você pode criar um programa de desenho, fazer arte interativa e animada ou criar, em termos programáticos, seus próprios elementos da interface do usuário, entre muitas possibilidades.

Mais tópicos da Ajuda

flash.display.Graphics

Noções básicas da API de desenho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

API de desenho é o nome da funcionalidade incorporada ao ActionScript que permite criar gráficos vetoriais (linhas, curvas, formas, preenchimentos e gradientes) e exibi-los na tela usando o ActionScript. A classe flash.display.Graphics oferece esta funcionalidade. É possível desenhar com o ActionScript em qualquer ocorrência de Shape, Sprite ou MovieClip usando a propriedade `graphics` definida em cada uma dessas classes. (A propriedade `graphics` de cada uma dessas classes é, na verdade, uma ocorrência da classe Graphics.)

Se você está apenas começando a desenhar com código, a classe Graphics inclui diversos métodos que ajudam a desenhar formas comuns, como círculos, elipses, retângulos e retângulos com cantos arredondados. É possível desenhá-las como linhas vazias ou formas preenchidas. Quando você precisar de funcionalidade mais avançada, a classe Graphics também inclui métodos para desenhar linhas e curvas Bézier quadráticas, que podem ser usadas junto com as funções de trigonometria da classe Math para criar qualquer forma desejada.

Os tempos de execução do Flash (como Flash Player 10 e o Adobe AIR 1.5 e as versões posteriores) adicionam mais uma API de desenho, que permite desenhar formas inteiras de modo programático usando um único comando. Depois que você estiver familiarizado com a classe Graphics e as tarefas incluídas em "Noções básicas do uso da API de desenho", passe para "Uso avançado da API de desenho" na página 239 para saber mais sobre esses recursos da API de desenho.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você vai encontrar ao usar a API de desenho:

Ponto de ancoragem Uma das duas extremidades de uma curva Bézier quadrática.

Ponto de controle O ponto que define a direção e o grau da curva de uma curva Bézier quadrática. A linha curva nunca atinge o ponto de controle, mas a linha faz uma curva como se estivesse sendo desenhada perto do ponto de controle.

Espaço de coordenadas O gráfico de coordenadas contido em um objeto de exibição, no qual seus elementos-filho são posicionados.

Preencher A parte interna sólida de uma forma que tem uma linha preenchida com cor ou uma forma inteira que não tem contorno.

Gradiente Cor que consiste em uma transição gradual de uma cor para uma ou mais cores (ao contrário de uma cor sólida).

Ponto Um local isolado em um espaço de coordenadas. No sistema de coordenadas bidimensional usado no ActionScript, um ponto é definido por seu local no eixo x e no eixo y (as coordenadas do ponto).

Curva Bézier quadrática Tipo de curva definido por uma fórmula matemática específica. Neste tipo de curva, a forma de uma curva é calculada com base nas posições dos pontos de ancoragem (as extremidades da curva) e em um ponto de controle que define o grau e a direção da curva.

Dimensionamento O tamanho de um objeto em relação ao seu tamanho original. Quando usado como um verbo, dimensionar um objeto significa alterar seu tamanho, aumentando ou diminuindo o objeto.

Traçado A parte do contorno de uma forma que tem uma linha preenchida com cor ou as linhas de uma forma sem preenchimento.

Traduzir Alterar as coordenadas de um ponto de um espaço de coordenadas para outro.

Eixo X O eixo horizontal no sistema de coordenadas bidimensional usado no ActionScript.

Eixo Y O eixo vertical no sistema de coordenadas bidimensional usado no ActionScript.

A classe Graphics

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cada objeto Shape, Sprite e MovieClip tem uma propriedade `graphics`, que é uma ocorrência da classe Graphics. A classe Graphics inclui propriedades e métodos para desenhar linhas, preenchimentos e formas. Se você quiser que o objeto display seja usado apenas como uma tela de desenho para o conteúdo do desenho, use uma ocorrência Shape. Uma ocorrência Shape será executada de modo melhor do que outros objetos display para desenho, porque ela não tem a sobrecarga da funcionalidade adicional nas classes Sprite e MovieClip. Se quiser um objeto display no qual seja possível desenhar conteúdo gráfico e no qual outros objetos display estejam contidos, use uma ocorrência Sprite. Para obter mais informações sobre a determinação dos objetos display que serão usados para várias tarefas, consulte "Escolha de uma subclasse de DisplayObject" na página 174.

Desenho de linhas e curvas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Todos os desenhos feitos com uma ocorrência Graphics baseiam-se em desenhos básicos com linhas e curvas. Consequentemente, todos os desenhos do ActionScript devem ser executados utilizando a mesma série de etapas:

- Definir estilos de linha e preenchimento
- Definir posição inicial do desenho
- Desenhar linhas, curvas e formas (opcionalmente movendo o ponto de desenho)
- Concluir a criação de um preenchimento, se necessário

Definição de estilos de linha e preenchimento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para desenhar com a propriedade `graphics` de uma ocorrência de Shape, Sprite ou MovieClip, defina primeiro o estilo (tamanho e cor da linha, cor do preenchimento) a ser usado no desenho. Da mesma forma como acontece quando você usa as ferramentas de desenho do Adobe® Flash® Professional ou outro aplicativo de desenho, ao usar o ActionScript para desenhar, você pode desenhar com ou sem traçado e com ou sem uma cor de preenchimento. Você especifica a aparência do traçado utilizando o método `lineStyle()` ou `lineGradientStyle()`. Para criar uma linha sólida, use o método `lineStyle()`. Quando você chamar esse método, os valores mais comuns que você especificará serão os três primeiros parâmetros: espessura da linha, cor e alfa. Por exemplo, essa linha de código instrui Shape chamada `myShape` a desenhar linhas com espessura de 2 pixels, vermelhas (0x990000) e 75% opacas:

```
myShape.graphics.lineStyle(2, 0x990000, .75);
```

O valor padrão para o parâmetro alpha é 1.0 (100%), então você pode deixá-lo desativado, se quiser uma linha totalmente opaca. O método `lineStyle()` também aceita dois parâmetros adicionais para dica de pixel e modo de escala. Para obter mais informações sobre o uso desses parâmetros, consulte a descrição do método `Graphics.lineStyle()` na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Para criar uma linha gradiente, use o método `lineGradientStyle()`. Esse método é descrito em "Criação de linhas e preenchimentos gradientes" na página 231.

Se você quiser criar uma forma preenchida, chame os métodos `beginFill()`, `beginGradientFill()`, `beginBitmapFill()` ou `beginShaderFill()` antes de iniciar o desenho. O mais básico deles, o método `beginFill()`, aceita dois parâmetros: a cor de preenchimento e (opcionalmente) um valor alfa para a cor de preenchimento. Por exemplo, se você quiser desenhar uma forma com um preenchimento verde sólido, utilize o seguinte código (considerando que está desenhando em um objeto chamado `myShape`):

```
myShape.graphics.beginFill(0x00FF00);
```

Chamar qualquer método de preenchimento implicitamente encerra qualquer preenchimento anterior antes de iniciar um novo. Chamar qualquer método que especifique um estilo de traçado substitui o traçado anterior, mas não altera o preenchimento especificado anteriormente e vice-versa.

Depois de especificar as propriedades style e fill da linha, a próxima etapa é indicar o ponto de início para o desenho. A ocorrência Graphics tem um ponto de desenho, como a ponta de uma caneta em um pedaço de papel. Sempre que o ponto de desenho for localizado, esse será o local em que a próxima ação de desenho será iniciada. Inicialmente um objeto Graphics começa com seu ponto de desenho no ponto 0, 0 no espaço de coordenadas do objeto em que está o desenho. Para iniciar o desenho em um ponto diferente, você pode primeiro chamar o método `moveTo()` antes de chamar um dos métodos de desenho. Isso é semelhante a tirar do papel a ponta da caneta e movê-la para uma nova posição.

Com o ponto de desenho no local, desenhe usando uma série de chamadas de métodos de desenho `lineTo()` (para linhas retas) e `curveTo()` (para linhas curvas).

Enquanto estiver desenhando, chame o método `moveTo()` a qualquer momento para mover o ponto de desenho para uma nova posição sem desenhar.

Ao desenhar, se você tiver especificado uma cor de preenchimento, feche o preenchimento chamando o método `endFill()`. Se você não desenhou uma forma fechada (em outras palavras, se no momento em que você chama `endFill()`, o ponto de desenho não está no ponto inicial da forma), quando chamar o método `endFill()`, o tempo de execução do Flash fecha automaticamente a forma desenhando uma linha reta do ponto de desenho atual até o local especificado na chamada mais recente de `moveTo()`. Se você iniciou um preenchimento e não chamou `endFill()`, chamar `beginFill()` (ou qualquer um dos outros métodos de preenchimento) fecha o preenchimento atual e inicia um novo.

Desenho de linhas retas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando você chama o método `lineTo()`, o objeto Graphics desenha uma linha reta do ponto de desenho atual até as coordenadas especificadas como os dois parâmetros na chamada do método, com o estilo de linha também especificado. Por exemplo, essa linha de código coloca o ponto de desenho no ponto 100, 100 e desenha uma linha até o ponto 200, 200:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.lineTo(200, 200);
```

O exemplo a seguir desenha triângulos vermelho e verde com uma altura de 100 pixels:

```
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);
```

Desenho de curvas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `curveTo()` desenha uma curva Bézier quadrática. Isso desenha um arco que conecta dois pontos (chamados pontos de ancoragem, enquanto se curva a um terceiro ponto (chamado ponto de controle). O objeto Graphics usa a posição atual do desenho como o primeiro ponto de ancoragem. Quando você chama o método `curveTo()`, passa quatro parâmetros: as coordenadas x e y do ponto de controle, seguido pelas coordenadas x e y do segundo ponto de ancoragem. Por exemplo, o seguinte código desenha uma curva começando no ponto 100, 100 e terminando no ponto 200, 200. Como o ponto de controle está no ponto 175, 125, isso cria uma curva que se move para a direita e para baixo:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.curveTo(175, 125, 200, 200);
```

O exemplo a seguir desenha objetos circulares vermelho e verde com uma largura e altura de 100 pixels. Observe que, devido à natureza da equação de Bézier de segundo grau, esses círculos não são perfeitos.

```
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);
```

Desenho de formas utilizando os métodos incorporados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para sua conveniência, ao desenhar formas comuns como círculos, elipses, retângulos e retângulos com cantos arredondados, o ActionScript 3.0 tem métodos que desenham essas formas comuns para você. Estes são os métodos `drawCircle()`, `drawEllipse()`, `drawRect()` e `drawRoundRect()` da classe Graphics. Esses métodos também podem ser utilizados no lugar dos métodos `lineTo()` e `curveTo()`. Observe, entretanto, que você ainda deve especificar os estilos de linha e preenchimento antes de chamar esses métodos.

O exemplo a seguir recria o exemplo de desenho de quadrados vermelho, verde e azul com largura e altura de 100 pixels. Esse código usa o método `drawRect()` e, além disso, especifica que a cor de preenchimento tem um alfa de 50% (0,5):

```
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);
```

Em um objeto Sprite ou MovieClip, o conteúdo de desenho criado com a propriedade `graphics` sempre aparece atrás de todos os objetos display filhos contidos no objeto. Além disso, o conteúdo da propriedade `graphics` não é um objeto display separado; portanto ele não aparece na lista de filhos do objeto Sprite ou MovieClip. Por exemplo, o seguinte objeto Sprite tem um círculo desenhado com sua propriedade `graphics` e um objeto TextField em sua lista de objetos display filhos:

```
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);
```

Observe que TextField aparece na parte superior do círculo desenhado com o objeto graphics.

Criação de linhas e preenchimentos gradientes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O objeto graphics também pode desenhar traçados e preenchimentos com gradientes em vez de cores sólidas. Um traçado gradiente é criado com o método `lineGradientStyle()` e um preenchimento de gradiente é criado com o método `beginGradientFill()`.

Os dois métodos aceitam os mesmos parâmetros. Os primeiros quatro são obrigatórios: tipo, cores, alfas e proporções. Os outros quatro são opcionais, mas úteis para personalização avançada.

- O primeiro parâmetro especifica o tipo de gradiente que está criando. Os valores aceitáveis são `GradientType.LINEAR` ou `GradientType.RADIAL`.
- O segundo parâmetro especifica a matriz dos valores de cor que serão utilizados. Em um gradiente linear, as cores serão organizadas da esquerda para a direita. Em um gradiente radial, as cores serão organizadas de dentro para fora. A ordem das cores da matriz representa a ordem em que elas serão desenhadas no gradiente.
- O terceiro parâmetro especifica os valores de transparência alfa das cores correspondentes no parâmetro anterior.
- O quarto parâmetro especifica as proporções ou a ênfase que cada cor tem no gradiente. A faixa de valores aceitável é de 0 a 255. Esses valores não representam a largura ou a altura, mas a posição no gradiente; 0 representa o início do gradiente e 255 representa o final do gradiente. A matriz de proporções deve aumentar sequencialmente e tem o mesmo número de entradas que as matrizes de cores e alfa especificadas no segundo e no terceiro parâmetros.

Embora o quinto parâmetro, a matriz de transformação, seja opcional, ele é normalmente usado porque fornece um modo fácil e eficiente de controlar a aparência do gradiente. Esse parâmetro aceita uma ocorrência Matrix. O modo mais fácil de criar um objeto Matrix para um gradiente é usar o método `createGradientBox()` da classe Matrix.

Definição de um objeto Matrix para usar com um gradiente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você usa os métodos `beginGradientFill()` e `lineGradientStyle()` da classe flash.display.Graphics para definir gradientes para usar em formas. Ao definir um gradiente, você fornece uma matriz como um dos parâmetros desses métodos.

A maneira mais fácil de definir a matriz é utilizando o método `createGradientBox()` da classe Matrix, que cria uma matriz utilizada para definir o gradiente. Você define a escala, a rotação e a posição do gradiente utilizando os parâmetros passados para o método `createGradientBox()`. O método `createGradientBox()` aceita os seguintes parâmetros:

- Largura da caixa de gradientes: a largura (em pixels) para a qual o gradiente será ampliado.
- Altura da caixa de gradientes: a altura (em pixels) para a qual o gradiente será ampliado.
- Rotação da caixa de gradientes: a rotação (em radianos) que será aplicada ao gradiente.
- Movimentação horizontal: a distância (em pixels) que o gradiente se deslocará horizontalmente.
- Movimentação vertical: a distância (em pixels) que o gradiente se deslocará verticalmente.

Por exemplo, considere um gradiente com as seguintes características:

- `GradientType.LINEAR`
- Duas cores, verde e azul, com a matriz de `proporções` definidas para `[0, 255]`.
- `SpreadMethod.PAD`
- `InterpolationMethod.LINEAR_RGB`

Os seguintes exemplos mostram gradientes nos quais o parâmetro `rotation` do método `createGradientBox()` é diferente conforme indicado, mas todas as configurações permanecem as mesmas:

`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;`	

Os seguintes exemplos mostram os efeitos de um gradiente linear verde para azul no qual os parâmetros `rotation`, `tx` e `ty` do método `createGradientBox()` são diferentes conforme indicado, mas todas as outras configurações permanecem as mesmas:

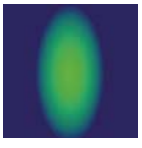
`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`	

`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;`	

Os parâmetros `width`, `height`, `tx` e `ty` do método `createGradientBox()` também afetam o tamanho e a posição de um preenchimento de gradiente *radial*, conforme mostram os exemplos a seguir:

`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 25;`
`ty = 0;`
	

O código a seguir produz o último gradiente radial ilustrado:

```
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);
```

Observe que a largura e a altura do preenchimento de gradiente são determinadas pela largura e altura da matriz de gradientes em vez da largura e altura desenhadas utilizando o objeto Graphics. Ao desenhar com objetos Graphics, você desenha o que existe naquelas coordenadas na matriz de gradiente. Mesmo que você use um dos métodos shape de um objeto Graphics como `drawRect()`, o gradiente não se amplia para o tamanho da forma que é desenhada — o tamanho do gradiente deve ser especificado na própria matrix de gradiente.

A seguir está uma ilustração da diferença visual entre as dimensões da matrix de gradiente e das dimensões do próprio desenho.

```
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);
```

Esse código desenha três gradientes com o mesmo estilo de preenchimento, especificado com uma distribuição igual de vermelho, verde e azul. Os gradientes são desenhados utilizando o método `drawRect()` com larguras de pixel de 50, 100 e 150 respectivamente. A matriz de gradiente que é especificada no método `beginGradientFill()` é criada com uma largura de 100 pixels. Isso significa que o primeiro gradiente engloba apenas meio espectro de gradiente, o segundo engloba todo ele e o terceiro engloba todo ele e tem 50 pixels adicionais de azul estendidos para a direita.

O método `lineGradientStyle()` funciona de modo semelhante a `beginGradientFill()` exceto pelo fato de que ao definir o gradiente, você deve especificar a espessura do traço utilizando o método `lineStyle()` antes do desenho. O código a seguir desenha uma caixa com um traçado gradiente vermelho, verde e azul:

```
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);
```

Para obter mais informações sobre a classe Matrix, consulte "Uso de objetos Matrix" na página 220.

Uso da classe Math com métodos de desenho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O objeto Graphics desenha círculos e quadrados, mas também pode desenhar formas mais complexas, especialmente quando os métodos de desenho são utilizados junto com as propriedades e os métodos da classe Math. A classe Math contém constantes matemáticas comuns, como `Math.PI` (aproximadamente 3.14159265...), uma constante para proporção de circunferência de um círculo em relação ao seu diâmetro. Ela também contém métodos para funções de trigonometria, incluindo `Math.sin()`, `Math.cos()` e `Math.tan()` entre outros. Desenhar formas utilizando esses métodos e constantes cria efeitos visuais mais dinâmicos, especialmente quando utilizados com repetição ou recursão.

Muitos métodos da classe Math esperam medidas circulares em unidades de radianos em vez de graus. A conversão entre esses dois tipos de unidades é um uso comum da classe Math:

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```

O exemplo a seguir cria uma onda senoidal e uma cossenoidal entre os métodos `Math.sin()` e `Math.cos()` para um determinado valor.

```
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);
}
```

Animação com a API de desenho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma vantagem da criação de conteúdo com a API de desenho é que você não está limitado a posicionar o conteúdo apenas uma vez. O que você desenha pode ser modificado pela manutenção e modificação das variáveis que usa ao desenhar. Você pode transmitir a animação alterando as variáveis e redesenhando, em um período de quadros ou com um temporizador.

Por exemplo, o código a seguir altera a exibição com cada quadro transmitido (atendendo ao evento `Event.ENTER_FRAME`), incrementando a contagem de graus atual, direciona o objeto graphics para limpar e redesenhar na posição atualizada.

```
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;
}
```

Para produzir um resultado significativamente diferente, você pode modificar as variáveis base iniciais no início do código, `currentDegrees`, `radius` e`satelliteRadius`. Por exemplo, tente reduzir a variável radius e/ou aumentar a variável totalSatellites. Esse é apenas um exemplo de como a API de desenho pode criar uma exibição visual cuja complexidade oculta a simplicidade de sua criação.

Exemplo de API de desenho: gerador visual algorítmico

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo do Gerador visual algorítmico desenha dinamicamente no palco vários "satélites" ou círculos que se movem em uma órbita circular. Entre os recursos explorados estão:

- Uso da API de desenho para desenhar uma forma básica com aparências dinâmicas
- Conexão da interação do usuário com as propriedades utilizadas em um desenho
- Transmissão de animação por meio da limpeza do palco em cada quadro e redesenho

O exemplo na subseção anterior animou um “satélite” solitário utilizando o evento `Event.ENTER_FRAME`. Esse exemplo vai além disso, criando um painel de controle com séries de controles deslizantes que atualizam imediatamente a exibição visual de vários satélites. Esse exemplo formaliza o código em classes externas e agrupa o código de criação de satélites em um loop, armazenando uma referência para cada satélite em uma matriz `satellites`.

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo podem ser encontrados na pasta Amostras/AlgorithmicVisualGenerator. Essa pasta contém os seguintes arquivos:

Arquivo	Descrição
AlgorithmicVisualGenerator.fla	O arquivo principal do aplicativo no Flash Professional (FLA).
com/example/programmingas3/algorithmic/AlgorithmicVisualGenerator.as	A classe que fornece a funcionalidade principal do aplicativo, que inclui desenho de satélites no palco e resposta aos eventos do painel de controle para atualizar as variáveis que afetam o desenho dos satélites.
com/example/programmingas3/algorithmic/ControlPanel.as	A classe que gerencia a interação do usuário com vários controles deslizantes e eventos de despacho quando eles ocorrem.
com/example/programmingas3/algorithmic/Satellite.as	Uma classe que representa o objeto display que gira em uma órbita ao redor de um ponto central e contém propriedades relacionadas ao estados atual do seu desenho.

Definição de ouvintes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo primeiro cria três ouvintes. O primeiro atende um evento despachado do painel de controle informando que é necessário recriar os satélites. O segundo atende às alterações de tamanho do palco do arquivo SWF. O terceiro atende a cada transmissão de quadro no arquivo SWF e para redesenhar utilizando a função `doEveryFrame()`.

Criação de satélites

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Assim que os ouvintes estiverem definidos, a função`build()` é chamada. Essa função chama primeiro a função `clear()`, que esvazia a matriz `satellites` e apaga qualquer desenho anterior no palco. Isso é necessário pois a função `build()` pode ser chamada novamente sempre que o painel de controle envia um evento para isso, como quando as configurações de cor são alteradas. Nesse caso, os satélites devem ser removidos e recriados.

A função então cria os satélites, configurando as propriedades iniciais necessárias para a criação, como a variável `position`, que começa em uma posição aleatória na órbita, e a variável `color`, que neste exemplo não é alterada pois o satélite foi criado.

Como cada satélite é criado, uma referência a ele é incluída na matriz `satellites`. Quando a função `doEveryFrame()` é chamada, ela será atualizada para todos os satélites nessa matriz.

Atualização da posição do satélite

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A função `doEveryFrame()` é o coração do processo de animação do aplicativo. Ela é chamada para cada quadro, em uma taxa igual à taxa de quadro do arquivo SWF. As pequenas alterações das variáveis do desenho transmitem a aparência da animação.

A função primeiro apaga todos os desenhos anteriores e redesenha o plano de fundo. Em seguida, ela percorre cada contêiner de satélite, incrementa a propriedade `position` de cada satélite e atualiza as propriedades `radius` e `orbitRadius` que podem ter sido alteradas na interação do usuário com o painel de controle. Por fim, o satélite atualiza-se para a sua nova posição chamando o método `draw()` da classe Satellite.

Observe que o contador, `i`, incrementa apenas até a variável `visibleSatellites`. Isso porque se o usuário limitar a quantidade de satélites que são exibidos no painel de controle, os satélites restantes no loop não deverão ser redesenhados, mas deverão ficar ocultos. Isso ocorre em um loop que segue imediatamente o loop responsável pelo desenho.

Quando a função doEveryFrame() é concluída, o número de `visibleSatellites` é atualizado na posição na tela.

Resposta à interação do usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A interação do usuário ocorre por meio do painel de controle, que é gerenciado pela classe ControlPanel. Essa classe define um ouvinte junto com os valores individuais mínimo, máximo e padrão de cada controle deslizante. À medida que o usuário move esses controles, a função `changeSetting()` é chamada. Essa função atualiza as propriedades do painel de controle. Se a alteração exigir a criação da exibição, um evento é despachado e então tratado no arquivo principal do aplicativo. À medida que as configurações do painel de controle são alteradas, a função `doEveryFrame()` desenha cada satélite com as variáveis atualizadas.

Personalização posterior

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Esse exemplo é apenas um esquema básico de como gerar visuais utilizando a API de desenho. Ele usa relativamente poucas linhas de código para criar uma experiência interativa que parece muito complexa. Apesar disso, esse exemplo pode ser estendido com pequenas alterações Algumas ideias:

- A função `doEveryFrame()` pode incrementar o valor de cor do satélite.
- A função `doEveryFrame()` pode reduzir ou expandir o raio do satélite com o tempo.
- O raio do satélite não precisa ser circular; ele pode usar a classe Math para se mover de acordo com uma onda senoidal, por exemplo.
- Os satélites podem usar a detecção de pressionamento com outros satélites.

A API de desenho pode ser utilizada como uma alternativa para criar efeitos visuais no ambiente de autoria do Flash, desenhando formas básicas no tempo de execução. Mas ela também pode ser utilizada para criar efeitos visuais diversos e com escopo diferente que não podem ser criados manualmente. Com a utilização da API de desenho e um pouco de matemática, o autor do ActionScript pode dar vida a várias criações inesperadas.

Uso avançado da API de desenho

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O Flash Player 10, o Adobe AIR 1.5 e os tempos de execução posteriores do Flash suportam um conjunto avançado de recursos de desenho. Os aprimoramentos da API de desenho para esses tempos de execução expandem os métodos de desenho de versões anteriores, por isso é possível estabelecer conjuntos de dados para gerar formas, alterar formas em tempo de execução e criar efeitos tridimensionais. Os aprimoramentos feitos na API de desenho consolidam os métodos existentes em comandos alternativos. Esses comandos utilizam matrizes de vetor e classes de enumeração para fornecer conjuntos de dados aos métodos de desenho. O uso de matrizes de vetor permite que formas mais complexas sejam renderizadas rapidamente e que os desenvolvedores alterem os valores de matriz de modo programático para a renderização dinâmica de formas em tempo de execução.

Os recursos de desenho introduzidos no Flash Player 10 estão descritos nas seguintes seções: "Caminhos de desenho" na página 240, "Definição de regras de contorno" na página 241, "Uso de classes de dados gráficos" na página 243 e "Sobre o uso de drawTriangles()" na página 246.

As seguintes tarefas são operações que provavelmente você executará usando a API de desenho avançada do ActionScript:

- Uso de objetos Vector para armazenar dados para métodos de desenho
- Definição de caminhos para desenhar formas de modo programático em uma só operação
- Definição de regras de contorno para determinar como preencher formas sobrepostas
- Ler o conteúdo dos gráficos vetoriais de um objeto de exibição, como para serializar e salvar os dados dos gráficos, gerar uma planilha de entidades gráficas em tempo de execução ou tirar uma cópia do conteúdo dos gráficos vetoriais.
- Uso de triângulos e métodos de desenho para efeitos tridimensionais

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes usados nesta seção:

- Vector: uma matriz de valores, todos com o mesmo tipo de dados. Um objeto Vector pode armazenar uma matriz de valores que os métodos de desenho usam para construir linhas e formas com um único comando. Para obter mais informações sobre objetos Vector, consulte "Matrizes indexadas" na página 27.
- Caminho: um caminho é formado por um ou mais segmentos retos ou curvos. O início e o final de cada segmento são marcados por coordenadas, que funcionam como alfinetes que prendem um esboço. Um caminho pode ser fechado (por exemplo, um círculo) ou aberto, com extremidades distintas (como uma linha ondulada).
- Contorno: a direção de um caminho conforme interpretada pelo renderizador, seja positiva (sentido horário) ou negativa (sentido anti-horário).
- GraphicsStroke: classe usada para definir o estilo da linha. Embora o termo "traçado" não faça parte dos aprimoramentos da API de desenho, o uso de uma classe para designar um estilo de linha com sua propriedade de preenchimento faz parte da nova API de desenho. É possível ajustar o estilo de uma linha dinamicamente usando a classe GraphicsStroke.
- Objeto Fill: objetos criados usando classes de exibição, como flash.display.GraphicsBitmapFill e flash.display.GraphicsGradientFill, que são passadas para o comando de desenho `Graphics.drawGraphicsData()`. Os objetos Fill e os comandos de desenho aprimorados introduzem uma abordagem de programação mais orientada a objetos para replicar `Graphics.beginBitmapFill()` e `Graphics.beginGradientFill()`.

Caminhos de desenho

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A seção sobre como desenhar linhas e curvas (consulte "Desenho de linhas e curvas" na página 227) apresentou os comandos usados para desenhar uma única linha (`Graphics.lineTo()`) ou curva (`Graphics.curveTo()`) e mover a linha até outro ponto (`Graphics.moveTo()`) para obter uma forma. Os métodos `Graphics.drawPath()` e `Graphics.drawTriangles()` aceitam um conjunto de objetos que representam os mesmos comandos de desenho como um parâmetro. Pelo uso desses métodos, você pode fornecer uma série de comandos `Graphics.lineTo()`, `Graphics.curveTo()` ou `Graphics.moveTo()` para que o runtime do Flash seja executado em uma única instrução.

A classe de enumeração GraphicsPathCommand define algumas constantes para comandos de desenho. Você transmite algumas dessas constantes (dispostas em uma instância de Vector) como um parâmetro para o método `Graphics.drawPath()`. Em seguida, com um único comando, você pode renderizar uma forma inteira ou várias formas. Também é possível alterar os valores transmitidos para esses métodos a fim de alterar uma forma existente.

Além do Vector de comandos de desenho, o método `drawPath()` precisa de algumas coordenadas que correspondam às coordenadas de cada comando de desenho. Crie uma instância de Vector contendo as coordenadas (ocorrências de Number) e transmita-a para o método `drawPath()` como o segundo argumento (`data`) .

Nota: *Os valores do vetor não são objetos Point; o vetor consiste em uma série de números em que cada grupo de dois números representa um par de coordenadas x/y.*

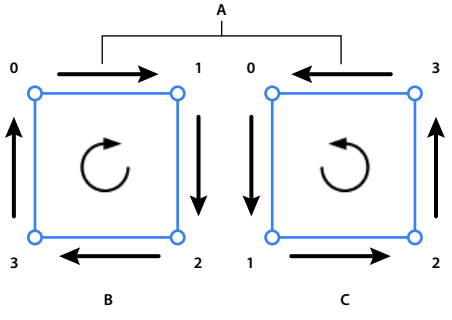
O método `Graphics.drawPath()` compara cada comando com seus respectivos valores de ponto (uma coleção de dois ou quatro números) para gerar um caminho no objeto Graphics:

```
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);
        }
    }
}
```

Definição de regras de contorno

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A API de desenho aprimorada também introduz o conceito de "espiral" do caminho: a direção para o caminho. O contorno de um caminho é tanto positivo (sentido horário) quanto negativo (sentido anti-horário). A ordem em que o renderizador interpreta as coordenadas fornecidas pelo vetor para o parâmetro de dados determina o contorno.



Contorno positivo e negativo
A. *Setas indicam a direção do desenho* ***B.*** *Rotação positiva (sentido horário)* ***C.*** *Rotação negativa (sentido anti-horário)*

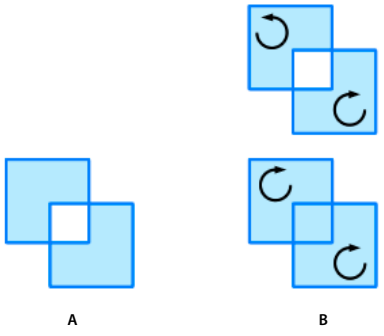
Além disso, observe que o método `Graphics.drawPath()` tem um terceiro parâmetro opcional, chamado “winding”:

```
drawPath(commands:Vector.<int>, data:Vector.<Number>, winding:String = "evenOdd"):void
```

Neste contexto, o terceiro parâmetro é uma string ou uma constante que especifica a regra de contorno ou de preenchimento para intersecção de caminhos. (Os valores de constante são definidos na classe GraphicsPathWinding como `GraphicsPathWinding.EVEN_ODD` ou `GraphicsPathWinding.NON_ZERO`.) A regra de contorno é importante quando ocorre intersecção de caminhos.

A regra par-ímpar é a regra de contorno padrão, sendo usada pela API de desenho herdada. Par-ímpar também é a regra padrão para o método `Graphics.drawPath()`. Com a regra de contorno par-ímpar, qualquer caminho de intersecção alterna entre preenchimentos abertos e fechados. Se houver intersecção de dois quadrados desenhados com o mesmo preenchimento, a área em que ocorre a intersecção será preenchida. Geralmente, as áreas adjacentes não são ambas preenchidas nem ambas não preenchidas.

A regra diferente de zero, por outro lado, depende do contorno (direção do desenho) para determinar se áreas definidas por caminhos de intersecção são preenchidas. Quando caminhos de contorno opostos se cruzam, a área definida não é preenchida, bem parecido com o que ocorre na regra par-ímpar. Para caminhos com o mesmo contorno, a área que não seria preenchida é preenchida:

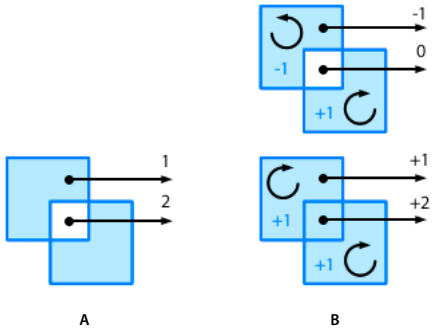


Regras de contorno para áreas de intersecção
A. *Regra de contorno par-ímpar* ***B.*** *Regra de contorno diferente de zero*

Nomes de regras de contorno

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Os nomes referem-se a uma regra mais específica que define como esses preenchimentos são gerenciados. Caminhos de rotação positiva recebem um valor de +1; caminhos de rotação negativa recebem um valor de -1. Começando em um ponto dentro de uma área fechada de uma forma, desenhe uma linha a partir desse ponto que se estenda indefinidamente. O número de vezes que a linha cruza um caminho e os valores combinados desses caminhos são usados para determinar o preenchimento. Para contorno par-ímpar, é usado o número de vezes em que a linha cruza um caminho. Quando a contagem é ímpar, a área é preenchida. Para contagens pares, a área não é preenchida. Para contorno diferente de zero, são usados os valores atribuídos aos caminhos. Quando os valores combinados do caminho não são 0, a área é preenchida. Quando os valores combinados são 0, a área não é preenchida.



Preenchimentos e contagens de regras de contorno
A. *Regra de contorno par-ímpar* ***B.*** *Regra de contorno diferente de zero*

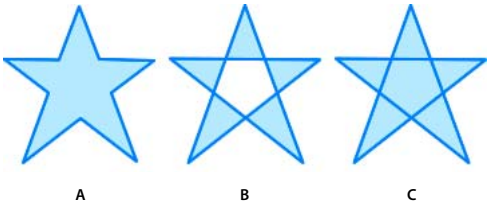
Uso de regras de contorno

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Essas regras de preenchimento são complicadas, mas em algumas situações elas são necessárias. Por exemplo, pense no desenho de uma forma de estrela. Com a regra par-ímpar padrão, a forma exigiria dez linhas diferentes. Com a regra de contorno diferente de zero, essas dez linhas são reduzidas a cinco. Este é o ActionScript de uma estrela com cinco linhas e uma regra de contorno diferente de zero:

```
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);
```

E esta é a forma de estrela:



Uma forma de estrela usando regras de contorno diferente
A. *10 linhas pares-ímpares* ***B.*** *5 linhas pares-ímpares* ***C.*** *5 linhas diferentes de zero*

E, à medida que imagens são animadas ou usadas como texturas em objetos tridimensionais e se sobrepõem, as regras de contorno tornam-se mais importantes.

Uso de classes de dados gráficos

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A API de desenho aprimorada inclui algumas classes no pacote flash.display que implementam a interface IGraphicsData. Essas classes funcionam como objetos de valor (contêineres de dados) que representam os métodos de desenho da API de desenho.

As seguintes classes implementam a interface IGraphicsData:

- GraphicsBitmapFill
- GraphicsEndFill

- GraphicsGradientFill
- GraphicsPath
- GraphicsShaderFill
- GraphicsSolidFill
- GraphicsStroke
- GraphicsTrianglePath

Com elas, é possível armazenar desenhos completos em um objeto Vector do tipo IGraphicsData (Vector.<IGraphicsData>). Você poderá, em seguida, reutilizar os dados de gráficos como a fonte de dados para outras instâncias de forma ou para armazenar informações de desenho para uso posterior.

Observe que existem várias classes de preenchimento para cada estilo de preenchimento, mas somente uma classe de traçado. O ActionScript tem apenas uma classe de traçado IGraphicsData, pois a classe de traçado usa as classes de preenchimento para definir seu estilo. Desta forma, todo traçado é, na verdade, definido por uma combinação da classe de traçado e de uma classe de preenchimento. Do contrário, a API dessas classes de dados gráficos espelha os métodos que elas representam na classe flash.display.Graphics:

Método de Graphics	**Classe correspondente**
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

Além disso, a classe GraphicsPath tem seus próprios métodos de utilitário `GraphicsPath.moveTo()`, `GraphicsPath.lineTo()`, `GraphicsPath.curveTo()`, `GraphicsPath.wideLineTo()` e `GraphicsPath.wideMoveTo()` que facilitam a definição desses comandos para uma instância de GraphicsPath. Esses métodos utilitários facilitam a tarefa de definir ou atualizar diretamente os comandos e valores de dados.

Desenhar com os dados dos gráficos vetoriais

Uma vez que você tenha uma coleção de instâncias de IGraphicsData, use o método `drawGraphicsData()` da classe Graphics para renderizar os gráficos. O método `drawGraphicsData()` executa algumas instruções de desenho de um vetor de instâncias de IGraphicsData em ordem sequencial:

```
// 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);
```

Modificando-se um valor no caminho usado pelo desenho do exemplo, é possível redesenhar a forma várias vezes para uma imagem mais complexa:

```
// 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);
```

Embora objetos IGraphicsData possam definir estilos de preenchimento e de traçado, esses estilos não são obrigatórios. Em outras palavras, os métodos da classe Graphics podem ser usados para definir estilos, enquanto os objetos IGraphicsData podem ser usados para desenhar uma coleção de caminhos salvos ou vice-versa.

Nota: *Use o método `Graphics.clear()` para remover um desenho anterior antes de começar um novo, a menos que você esteja aumentando o desenho original, como visto no exemplo acima. Quando alterar uma única parte de um caminho ou de uma coleção de objetos IGraphicsData, redesenhe o desenho inteiro para ver as alterações.*

Quando são usadas classes de dados gráficos, o preenchimento é renderizado sempre que três ou mais pontos são desenhados porque a forma é inerentemente fechada nesse ponto. Embora o preenchimento seja fechado, o traçado não é, e esse comportamento é diferente de quando são usados vários comandos `Graphics.lineTo()` ou `Graphics.moveTo()`.

Leitura dos dados dos gráficos vetoriais

Flash Player 11.6 e posterior, Adobe AIR 3.6 e posterior

Além de desenhar o conteúdo do vetor para o objeto de exibição, no Flash Player 11.6 e Adobe AIR 3.6 e posterior, você pode utilizar o método `readGraphicsData()` da classe Graphics para obter uma representação de dados do conteúdo dos gráficos vetoriais de um objeto de exibição. Este método pode ser usado para gerar um instantâneo de um gráfico para salvar, copiar, criar uma planilha de entidades gráficas em runtime, entre outros.

Chamar o método `readGraphicsData()` retorna uma instância de Vector contendo objetos de IGraphicsData. Eles são os mesmos objetos utilizados para desenhar gráficos vetoriais com o método `drawGraphicsData()`.

Há várias limitações para a leitura de gráficos vetoriais com o método `readGraphicsData()`. Para obter mais informações, consulte a entrada readGraphicsData() na Referência de linguagem do ActionScript.

Sobre o uso de drawTriangles()

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Outro método avançado introduzido no Flash Player 10 e no Adobe AIR 1.5, `Graphics.drawTriangles()`, é como o método `Graphics.drawPath()`. O método `Graphics.drawTriangles()` também usa um Vector.<Number> para especificar localizações de pontos para desenhar um caminho.

No entanto, a verdadeira finalidade do método `Graphics.drawTriangles()` é facilitar os efeitos tridimensionais através do ActionScript. Para obter informações sobre como usar `Graphics.drawTriangles()` para produzir efeitos tridimensionais, consulte “Uso de triângulos para obter efeitos 3D” na página 373.

Capítulo 13: Trabalho com bitmaps

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Além dos recursos de desenho de vetores, o ActionScript 3.0 permite criar imagens de bitmap ou manipular dados em pixels de imagens de bitmap externas carregadas em um arquivo SWF. Com a habilidade de acessar e alterar valores de pixel individuais, você pode criar seus próprios efeitos de imagem semelhantes a filtros e usar as funções internas de ruído para criar texturas e ruídos aleatórios.

- Renaun Erickson: Renderização de recursos de jogos no ActionScript usando técnicas de blitting
- Programação de bitmap: Capítulo 26 de O Essencial do ActionScript 3, por Colin Moock (O'Reilly Media, 2007)
- Mike Jones: Trabalhando com Sprites no Mecanismo de Botões de Comando
- Flash e Malhas: Simplificando as Partículas de Pixel
- Flixel

Noções básicas do trabalho com bitmaps

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando você trabalha com imagens digitais, provavelmente encontra dois tipos principais de gráficos: bitmap e vetor. Os gráficos bitmap, também conhecidos como gráficos raster, são compostos por pequenos quadrados (pixels) organizados em um formato de grade retangular. Os gráficos de vetor são compostos por formas geométricas geradas matematicamente, como linhas, curvas e polígonos.

As imagens de bitmap são definidas pela largura e altura da imagem, medidas em pixels, e pelo número de bits contido em cada pixel, que representa o número de cores que um pixel pode conter. No caso das imagens de bitmap que utilizam o modelo de cor RGB, os pixels são compostos por três bytes: vermelho, verde e azul. Cada um desses bytes contém um valor que varia de 0 a 255. Quando os bytes são combinados em pixels, eles produzem uma cor semelhante à cor produzida quando um artista mistura cores de pintura. Por exemplo, um pixel que contém valores de byte de vermelho-255, verde-102 e azul-0 produz uma cor laranja vibrante.

A qualidade de uma imagem de bitmap é determinada pela combinação de resolução de imagem e o valor de bits da densidade de cor. *Resolução* relaciona-se ao número de pixels contido dentro de uma imagem. Quanto maior o número de pixels, mais alta é a resolução e melhor a imagem aparece. *Densidade de cor* relaciona-se à quantidade de informações que um pixel pode conter. Por exemplo, uma imagem que tem um valor de densidade de cor de 16 bits por pixel não pode representar o mesmo número de cores que uma imagem que tem uma densidade de cor de 48 bits. Consequentemente, a imagem de 48 bits terá graus de sombreamento mais suaves do que a contraparte de 16 bits.

Como os gráficos de bitmap dependem da resolução, eles não são dimensionados muito bem. Isso é mais facilmente notado quando imagens de bitmap são aumentadas em tamanho. Aumentar um bitmap normalmente resulta em perda de detalhes e qualidade.

Formatos de arquivo bitmap

As imagens bitmap são agrupadas em vários formatos de arquivo comuns. Esses formatos usam tipos diferentes de algoritmos de compactação para reduzir o tamanho do arquivo, bem como otimizar a qualidade da imagem com base no objeto final da imagem. Os formatos de imagem de bitmap suportados pelos tempos de execução da Adobe são BMP, GIF, JPG, PNG e TIFF.

BMP

O formato BMP (bit mapped) é um formato de imagem padrão usado pelo sistema operacional Microsoft Windows. Por não utilizar algoritmos de compactação de nenhum tipo, o tamanho dos arquivos BMP é maior.

GIF

O GIF (Graphics Interchange Format) foi originalmente desenvolvido pela CompuServe em 1987 como uma forma de transmitir imagem com 256 cores (cor de 8 bits). O formato fornece tamanhos pequenos de arquivo e é ideal para imagens com base na Web. Devido a esse palete de cores limitado do formato, as imagens GIF não são normalmente para fotografias, que geralmente exigem mais graus de sombreamento e gradientes de cor. As imagens GIF permitem transparência de bit único, possibilitando que as cores sejam mapeadas como claras (ou transparentes). Isso resulta na cor de plano de fundo de uma página da Web mostrada através da imagem onde a transparência foi mapeada.

JPEG

Desenvolvido pelo Joint Photographic Experts Group (JPEG), o formato de imagem JPEG (também escrito como JPG) usa um algoritmo de compactação com perdas para permitir densidade de cores de 24 bits com tamanho de arquivo pequeno. Compactação com perdas significa que sempre que a imagem é salva, ela perde qualidade de dados, mas resulta em um tamanho de arquivo menor. O formato JPEG é ideal para fotografias porque ele é capaz de exibir milhões de cores. A habilidade de controlar o grau de compactação aplicada a uma imagem permite manipular a qualidade da imagem e o tamanho do arquivo.

PNG

O formato PNG (Portable Network Graphics) foi produzido como uma alternativa de fonte aberta para o formato de arquivo GIF patenteado. O formato PNG suporta densidade de cores de até 64 bits, permitindo até 16 milhões de cores. Como o PNG é relativamente um novo formato, alguns navegadores antigos não suportam arquivos PNG. Diferentemente dos JPGs, os PNGs usam compactação sem perda, o que significa que nenhum dado da imagem é perdido quando a imagem é salva. Os arquivos PNG também suportam transparência alfa, que permite até 256 níveis de transparência.

TIFF

O TIFF (Tagged Image File Format) foi o formato compatível com várias plataformas escolhido antes do surgimento do PNG. A desvantagem desse formato é que, devido à grande variedade de TIFFs, não há um único leitor que possa manipular cada versão. Além disso, não há navegadores da Web que suportam esse formato atualmente. Esse formato usa compactação com ou sem perdas, e pode manipular espaços de cor específicos do dispositivo (como CMYK).

Bitmaps transparentes e opacos

As imagens de bitmap que usam os formatos GIF ou PNG podem ter um byte extra (canal alfa) adicionado a cada pixel. O byte de pixel extra representa o valor da transparência do pixel.

As imagens GIF permitem transparência de único bit, o que significa que você pode especificar uma única cor, de um palete de 256 cores, para ser transparente. As imagens PNG, por outro lado, podem ter até 256 níveis de transparência. Essa função traz benefícios especialmente quando imagens ou texto devem ficar mesclados nos planos de fundo.

O ActionScript 3.0 replica esse byte de pixel de transparência extra dentro da classe BitmapData. Semelhante ao modelo de transparência do PNG, o ActionScript oferece até 256 níveis de transparência.

Conceitos e termos importantes

A lista a seguir contém termos importantes que você vai encontrar ao conhecer o gráfico de bitmap:

Alfa O nível de transparência (ou mais precisamente, de opacidade) em uma cor ou uma imagem. O valor de alfa é frequentemente descrito como o valor do *canal alfa*.

Cor ARGB Um esquema de cores em que cada cor de pixel é uma mistura de valores das cores vermelho, verde e azul, e sua transparência é especificada como um valor alfa.

Canal de cor Normalmente, as cores são representadas como uma mistura de algumas cores básicas - geralmente (para gráficos de computador) vermelho, verde e azul. Cada cor básica é considerada um canal de cor; as quantidades de cor em cada canal de cor, misturadas, determinam a cor final.

Profundidade de cores Também conhecido como *densidade de bits*, refere-se à quantidade de memória do computador que está atribuída a cada pixel, que por sua vez determina o número de cores possíveis que podem ser representadas na imagem.

Pixel A menor unidade de informação em uma imagem de bitmap – essencialmente um ponto de cor.

Resolução As dimensões em pixel de uma imagem, que determinam o nível de detalhes granulados contidos na imagem. A resolução é frequentemente expressa em termos de largura e altura em número de pixels.

Cor RGB Um esquema de cores em que cada cor de pixel é representada como uma mistura de valores das cores vermelho, verde e azul.

Classes Bitmap e BitmapData

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As principais classes do ActionScript 3.0 para trabalhar com imagens de bitmpa são a classe Bitmap, que é usada para exibir as imagens de bitmap na tela, e a classe BitmapData, usada para acessar e manipular os dados brutos de imagem de um bitmpa.

Mais tópicos da Ajuda

flash.display.Bitmap

flash.display.BitmapData

Compreensão da classe Bitmap

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Como uma subclasse da classe DisplayObject, a classe Bitmap é a principal classe do ActionScript 3.0 usada para exibir imagens de bitmap. Essas imagens podem ter sido carregadas através da classe flash.display.Loader ou criada dinamicamente com o construtor `Bitmap()`. Durante o carregamento de uma imagem a partir de uma fonte externa, um objeto Bitmap pode usar apenas imagens nos formatos GIF, JPEG ou PNG. Depois de instanciado, a ocorrência de Bitmap pode ser considerada um wrapper para um objeto BitmapData que precisa ser renderizado no Palco. Como uma ocorrência de bitmap é um objeto de exibição, todos os recursos e funcionalidade dos objetos de exibição também podem ser usados para manipular uma ocorrência de Bitmap. Para obter mais informações sobre como trabalhar com objetos de exibição, consulte "Programação de exibição" na página 153.

Encaixe e suavização de pixels

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Além da funcionalidade comum para todos os objetos de exibição, a classe Bitmap fornece alguns recursos adicionais que são específicos para imagens bitmaps.

A propriedade `pixelSnapping` da classe Bitmap determina se um objeto Bitmap se encaixa ou não no seu pixel mais próximo. Essa propriedade aceita uma das três constantes definidas na classe PixelSnapping: `ALWAYS`, `AUTO` e`NEVER`.

A sintaxe para aplicação de encaixe de pixel como se segue:

```
myBitmap.pixelSnapping = PixelSnapping.ALWAYS;
```

Frequentemente, quando imagens de bitmap são escalada, elas ficam borradas e distorcidas. Para ajudar a reduzir essa distorção, use a propriedade `smoothing` da classe BitmapData. Essa propriedade Boolean, quando definida como `true`, suaviza os pixels dentro da imagem quando ela é escalada. Isso dá à imagem uma aparência mais clara e natural.

Compreensão da classe BitmapData

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe BitmapData, que está no pacote flash.display, pode ser comparada a um snapshot fotográfico de pixels contido em um imagem de bitmap carregada ou criada dinamicamente. Esse snapshot é representado por uma matriz de dados de pixel dentro do objeto. A classe BitmapData também contém uma série de métodos incorporados que são utilizados para a criação e a manipulação de dados de pixel.

Para instanciar um objeto BitmapData, use o seguinte código:

```
var myBitmap:BitmapData = new BitmapData(width:Number, height:Number, transparent:Boolean,
fillColor:uinit);
```

Os parâmetros `width` e `height` especificam o tamanho do bitmap. A partir do AIR 3 e do Flash player 11, os limites de tamanho de um objeto BitmapData foram removidos. O tamanho máximo de um bitmap depende do sistema operacional.

No AIR 1.5 e no Flash Player 10, o tamanho máximo de um objeto BitmapData é de 8.191 pixels de largura ou de altura, e o número total de pixels não pode exceder 16.777.215 pixels. (Dessa forma, caso um objeto BitmapData tenha 8.191 pixels de largura, ele só pode ter 2.048 pixels de altura.) No Flash Player 9 e anteriores e no AIR 1.1 e anteriores, a limitação é de 2.880 de altura e de 2.880 pixels de largura.

O parâmetro `transparent` especifica se os dados de bitmap incluem um canal alfa (`true`) ou não (`false`). O parâmetro `fillColor` é um valor de cor de 32 bits que especifica a cor de plano de fundo, bem como o valor de transparência (se ele tiver sido definido como `true`). O exemplo a seguir cria um objeto BitmapData com plano de fundo laranja que é 50% transparente.

```
var myBitmap:BitmapData = new BitmapData(150, 150, true, 0x80FF3300);
```

Para renderizar um objeto BitmapData criado recentemente na tela, atribua-o ou envolva-o em uma ocorrência de Bitmap. Para fazer isso, você pode transmitir o objeto BitmapData como um parâmetro do construtor de objeto Bitmap ou atribuí-lo à propriedade `bitmapData` de uma ocorrência de Bitmap existente. Você também deve adicionar a ocorrência de Bitmap à lista de exibição chamando os métodos `addChild()` ou `addChildAt()` do contêiner de objeto de exibição que conterá a ocorrência de Bitmap. Para obter mais informações sobre como trabalhar com a lista de exibição, consulte "Adição de objetos de exibição à lista de exibição" na página 161.

O exemplo a seguir cria um objeto BitmapData com preenchimento vermelho e exibe-o em uma ocorrência de Bitmap.

```
var myBitmapDataObject:BitmapData = new BitmapData(150, 150, false, 0xFF0000);
var myImage:Bitmap = new Bitmap(myBitmapDataObject);
addChild(myImage);
```

Manipulação de pixels

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe BitmapData também contém um conjunto de métodos que permitem manipular valores de dados de pixels.

Manipulação de pixels individuais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao alterar a aparência de uma imagem de bitmap no nível de pixel, você primeiro precisa obter os valores de cores dos pixels contidos dentro da área que deseja manipular. Use o método `getPixel()` para ler esses valores de pixel.

O método `getPixel()` recupera um valor RGB a partir de um conjunto de coordenadas x, y (pixel) transmitidas como parâmetros. Se qualquer um dos pixels que você deseja manipular incluir informações sobre transparência (canal alfa), será necessário usar o método `getPixel32()`. Esse método também recupera um valor RGB, mas diferente de `getPixel()`, o valor retornado por `getPixel32()` contém dados adicionais que representam o valor do canal alfa (transparência) do pixel selecionado.

Como alternativa, se você quiser simplesmente alterar a cor ou a transparência de um pixel contido em um bitmap, poderá usar o método `setPixel()` ou `setPixel32()`. Para definir uma cor de pixels, transmita simplesmente as coordenadas x, y e o valor da cor para um desses métodos.

O exemplo a seguir usa o método `setPixel()` para desenhar uma cruz em um plano de fundo BitmapData verde: Ele então usa `getPixel()` para recuperar o valor da cor do pixel nas coordenadas 50, 50 e liga o valor retornado.

```
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));
```

Se você quiser ler o valor de um grupo de pixels, use o método `getPixels()`. Esse método gera uma matriz de bytes a partir de uma região retangular dos dados de pixels que são transmitidos como um parâmetro. Cada um dos elementos da matriz de bytes (em outra palavras, os valores de pixel) são valores de pixel inteiros não assinados de 32 bits e não multiplicados.

Inversamente, para alterar (ou definir) o valor de um grupo de pixels, use o método `setPixels()`. Esse método espera dois parâmetros (`rect` e `inputByteArray`), que são combinados para resultar em uma região retangular (`rect`) dos dados de pixel (`inputByteArray`).

Como os dados são lidos (e gravados) fora de `inputByteArray`, o método `ByteArray.readUnsignedInt()` é chamado para cada um dos pixels na matriz. Se, por alguma razão, `inputByteArray` não contiver pelo menos um retangular inteiro de dados de pixel, o método interromperá o processamento dos dados da imagem naquele ponto.

É importante lembrar que, para obter e configurar dados de pixel, a matriz de byte espera valores de pixel vermelho, verde, azul e alfa de 32 bits (ARGB).

O exemplo a seguir usa os métodos `getPixels()` e `setPixels()` para copiar um grupo de pixels de um objeto BitmapData para outro:

```
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;
```

Detecção de colisão de nível de pixel

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `BitmapData.hitTest()` executa a detecção de colisão no nível de pixel entre os dados de bitmap e outro objeto ou ponto.

O método `BitmapData.hitTest()` aceita cinco parâmetros:

- `firstPoint` (Point): Esse parâmetro se refere à posição do pixel do canto superior esquerdo do primeiro BitmapData no qual o teste de ocorrência está sendo executado.
- `firstAlphaThreshold` (uint): Esse parâmetro especifica o valor de canal alfa mais alto que é considerado opaco para esse teste de ocorrência.
- `secondObject` (Object): Esse parâmetro representa a área de impacto. O objeto `secondObject` não pode ser um objeto Rectangle, Point, Bitmap ou BitmapData. Esse objeto representa a área de ocorrência na qual a detecção de colisão está sendo executada.
- `secondBitmapDataPoint` (Point): Esse parâmetro opcional é usado para definir uma localização de pixel no segundo objeto BitmapData. Use esse parâmetro apenas quando o valor de `secondObject` for um objeto BitmapData. O padrão é `null`.
- `secondAlphaThreshold` (uint): Esse parâmetro opcional representa o valor de canal alfa mais alto que é considerado opaco no segundo objeto BitmapData. O valor padrão é 1. Use esse parâmetro apenas quando o valor de `secondObject` for um objeto BitmapData e quando ambos os objetos BitmapData forem transparentes.

Ao executar a detecção de colisão em imagens opacas, lembre-se de que o ActionScript trata a imagem como se ela fosse um retângulo totalmente opaco (ou caixa delimitadora). Como alternativa, ao executar teste de ocorrência em nível de pixel em imagens transparentes, ambas as imagens devem ser transparentes. Além disso, o ActionScript usa os parâmetros de limitação de alfa para determinar em qual ponto os pixels são alterados de transparente para opaco.

O exemplo a seguir cria três imagens de bitmap e verifica se há colisão de pixel utilizando dois diferentes pontos de colisão (um retorna false e outro 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
```

Cópia de dados de bitmap

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para copiar dados bitmap de uma imagem para a outra, você pode usar vários métodos: `clone()`, `copyPixels()`, `copyChannel()`, `draw()` e `drawWithQuality()` (método `drawWithQuality` disponível no Flash Player 11.3 e superior; AIR 3.3 e superior).

Como o nome sugere, o método `clone()` permite clonar, ou obter uma amostra, dos dados de bitmap de um objeto BitmapData para outro. Quando chamado, o método retorna um novo objeto BitmapData que é um clone exato da ocorrência original do qual foi copiado.

O exemplo a seguir clona uma cópia de um quadrado (pai) laranja e coloca o clone ao lado do quadrado pai original:

```
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;
```

O método `copyPixels()` é um modo fácil e rápido de copiar pixels de um objeto BitmapData para outro. O método obtém um snapshot retangular (definido pelo parâmetro `sourceRect`) da imagem de origem e a copia para outra área retangular (de tamanho igual). O local do retângulo "colado" recentemente é definido dentro do parâmetro `destPoint`.

O método `copyChannel()` fornece amostras de um valor de canal de cor predefinido (alfa, vermelho, verde ou azul) de um objeto BitmapData de origem e o copia para um canal de um objeto BitmapData de destino. Chamar esse método não afeta os outros canais no objeto BitmapData de destino.

Os métodos `draw()` e `drawWithQuality()` desenham ou renderizam o conteúdo gráfico de um sprite de origem, clipe de filme ou outro objeto de exibição em um novo bitmap. Usando `matrix`, `colorTransform`, `blendMode` e os parâmetros de destino`clipRect`, você pode modificar a forma na qual o novo bitmap é renderizado. Esse método usa o processador de vetor no Flash Player e no AIR para gerar os dados.

Ao chamar `draw()` ou `drawWithQuality()`, você passa o objeto de origem (sprite, clipe de filme ou outro objeto de exibição) como o primeiro parâmetro, como demonstrado aqui:

```
myBitmap.draw(movieClip);
```

Se o objeto de origem teve qualquer transformação (cor, matriz etc.) aplicada a ele depois que ele foi originalmente carregado, essas transformações não serão copiadas para o novo objeto. Se você quiser copiar as transformações para o novo bitmap, será necessário copiar o valor da propriedade `transform` do objeto original para a propriedade `transform` do objeto Bitmap que usa o novo objeto BitmapData.

Compactando dados de bitmap

Flash Player 11.3 e posterior, AIR 3.3 e posterior

O método `flash.display.BitmapData.encode()` permite a compressão nativa de dados bitmap em um dos formatos de compactação de imagem a seguir:

- **PNG** - usa compactação de PNG, usando opcionalmente a compactação rápida, o que enfatiza a velocidade de compressão sobre o tamanho do arquivo. Para usar a compactação PNG, passe um novo objeto `flash.display.PNGEncoderOptions` como o segundo parâmetro do método `BitmapData.encode()`.
- **JPEG** - usa compactação de JPEG, especificando opcionalmente a qualidade da imagem. Para usar a compactação de JPEG, passe um novo objeto `flash.display.JPEGEncoderOptions` como o segundo parâmetro do método `BitmapData.encode()`.

- **JPEGXR** - usa a compactação JPEG Extended Range (XR), especificando opcionalmente as configurações do canal de cores, com perdas e de entropia. Para usar a compactação JPEGXR, passe um novo objeto `flash.display.JPEGXREncoderOptions` como o segundo parâmetro do método `BitmapData.encode()`.

Você pode usar esse recurso para processar imagens como parte de um carregamento do servidor ou fluxo de trabalho de download.

O snippet de exemplo a seguir compacta um objeto BitmapData usando `JPEGEncoderOptions`:

```
// 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);
```

Texturas com funções de ruído

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para modificar a aparência de um bitmap, você pode aplicar um efeito de ruído nele, usando o método `noise()` ou os métodos `perlinNoise()`. Um efeito de ruído pode ser comparado a um estático que aparece em uma tela de televisão não sintonizada.

Para aplicar um efeito de ruído a um bitmap, use o método `noise()`. Esse método aplica um valor de cor aleatório para pixels dentro de uma área especificada de uma imagem de bitmap.

Esse método aceita cinco parâmetros:

- `randomSeed` (int): O número base aleatório que determina o padrão. Apesar do nome, esse número cria realmente os mesmos resultados se o mesmo número é transmitido. Para obter um resultado aleatório verdadeiro, use o método `Math.random()` para transmitir um número aleatório para esse parâmetro.
- `low` (uint): Esse parâmetro se refere ao valor mais baixo a ser gerado para cada pixel (0 a 255). O valor padrão é 0. A configuração desse valor mais baixo resulta em um padrão de ruído mais escuro, enquanto a configuração de um valor mais alto resulta em um padrão mais claro.
- `high` (uint): Esse parâmetro se refere ao valor mais alto a ser gerado para cada pixel (0 a 255). O valor padrão é 255. A configuração desse valor mais baixo resulta em um padrão de ruído mais escuro, enquanto a configuração de um valor mais alto resulta em um padrão mais claro.
- `channelOptions` (uint): Esse parâmetro especifica a qual canal de cores do objeto Bitmap o padrão de ruído será aplicado. O número pode ser uma combinação de qualquer um dos quatro valores ARGB do canal de cores. O valor padrão é 7.
- `grayScale` (Boolean): Quando definido para `true`, esse parâmetro aplica o valor `randomSeed` aos pixels de bitmap, eliminando efetivamente todas as cores da imagem. O canal alfa não é afetado por esse parâmetro. O valor padrão é `false`.

O exemplo a seguir cria uma imagem de bitmap e aplica um padrão de ruído azul a ela:

```
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);
        }
    }
}
```

Se você quiser criar uma textura com a aparência mais orgânica, use o método `perlinNoise()` . O método `perlinNoise()` produz texturas orgânicas realistas que são ideais para fumaça, nuvens, água, fogo ou até mesmo explosões.

Como isso é gerado por um algoritmo, o método `perlinNoise()` usa menos memória do que as texturas com base em bitmap. Entretanto, ele pode ainda ter um impacto no uso do processador, tornando o conteúdo mais lento e fazendo com que a tela seja redesenhada mais lentamente do que a taxa de quadro, especialmente em computadores mais antigos. Isso deve-se principalmente aos cálculos de ponto de flutuação que precisam ocorrer para processar os algoritmos de ruído perlin.

O método aceita nove parâmetros (os primeiro seis são obrigatório):

- `baseX` (Number): Determina o valor de x (tamanho) dos padrões criados.
- `baseY` (Number): Determina o valor de y (tamanho) dos padrões criados.
- `numOctaves` (uint): Número de oitavas ou funções de ruído individuais a serem combinadas para criar esse ruído. Números maiores de oitavas criam imagens com mais detalhes, mas também exigem mais tempo de processamento.
- `randomSeed` (int): O número base aleatório funciona exatamente da mesma forma que na função `noise()`. Para obter um resultado aleatório verdadeiro, use o método `Math.random()` para transmitir um número aleatório para esse parâmetro.
- `stitch` (Boolean): Se definido para `true`, esse método tentará suavizar as bordas de transição da imagem para criar texturas contínuas para colocação lado a lado como um preenchimento de bitmap.
- `fractalNoise` (Boolean): Esse parâmetro relaciona as bordas de gradientes sendo geradas pelo método. Se definido para `true`, o método gerará ruído fractal que suaviza as bordas do efeito. Se definido como `false`, ocorre turbulência. Uma imagem com turbulência tem descontinuidades visíveis no gradiente que podem fazer com que ela aproxime melhor os efeitos visuais mais nítidos, como chamas e ondas do mar.
- `channelOptions` (uint): O parâmetro `channelOptions` funciona exatamente da mesma forma que no método `noise()`. Ele especifica a qual canal de cores (do bitmap) o padrão de ruído será aplicado. O número pode ser uma combinação de qualquer um dos quatro valores ARGB do canal de cores. O valor padrão é 7.
- `grayScale` (Boolean): O parâmetro `grayScale` funciona exatamente da mesma forma que no método `noise()`. Quando definido como `true`, esse parâmetro aplica o valor `randomSeed` aos pixels de bitmap, eliminando todas as cores da imagem de modo eficaz. O valor padrão é `false`.

- `deslocamentos` (Matriz): Uma matriz de pontos que corresponde a deslocamentos x e y para cada oitava. Ao manipular os valores de deslocamento, você pode rolar suavemente as camadas de uma imagem. Cada ponto na matriz de deslocamento afeta uma função de ruído de oitava específica. O valor padrão é `null`.

O exemplo a seguir cria um objeto BitmapData de 150 x 150 pixels que chama o método `perlinNoise()` para gerar um efeito de nuvem em verde e azul:

```
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);
        }
    }
}
```

Rolagem de bitmaps

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Imagine que você criou um aplicativo de mapeamento de ruas em que sempre que o usuário move o mapa você precisa atualizar a visualização (mesmo que o mapa tenha sido movido apenas alguns pixels).

Uma forma de criar essa funcionalidade seria reprocessar uma nova imagem contendo a visualização atualizada do mapa sempre que o usuário move o mapa. Como alternativa, você pode criar uma única imagem grande e usar o método `scroll()`.

O método `scroll()` copia um bitmap na tela e o cola em um local de deslocamento — especificado por parâmetros (`x`, `y`). Se acontecer de uma parte do bitmap ficar fora do palco, o efeito resultante indicará que a imagem foi deslocada. Quando combinado com uma função de temporizador (ou um evento `enterFrame`), você poderá fazer com a imagem pareça ser animada ou estar em rolagem.

O exemplo a seguir pega o exemplo de ruído perlin anterior e gera uma imagem de bitmap maior (três quartos dela são renderizados fora do palco). O método `scroll()` é então aplicado junto com um ouvinte do evento `enterFrame` que desloca a imagem por um pixel na direção diagonal para baixo. Esse método é chamado sempre que o quadro é inserido e, consequentemente, as partes da imagem fora da tela são renderizadas no Palco à medida que usamos a rolagem na imagem.

```
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);
}
```

Benefícios do mapeamento mip

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Mapas MIP (também conhecido como *mipmaps*), são bitmaps agrupados e associados a uma textura para aumentar a qualidade e o desempenho da renderização no tempo de execução. Cada imagem bitmap no mapa MIP é uma versão da imagem bitmap principal, mas a um nível de detalhes reduzido da imagem principal.

Por exemplo, você pode ter um mapa MIP que inclui na maior qualidade uma imagem principal de 64 × 64 pixels. As imagens de qualidade inferior no mapa MIP teriam 32 × 32, 16 × 16, 8 × 8, 4 × 4, 2 × 2 e 1 × 1 pixels.

O *Fluxo de textura* e a capacidade de carrega o de menor qualidade primeiro e depois exibir progressivamente bitmaps com maior conforme são carregados. Como os bitmaps de qualidade inferior são pequenos, ele carregam mais rápido do que a imagem principal. Portanto, os usuários dos aplicativos podem visualizar imagens em um aplicativo antes do principal de alta qualidade carregar.

O Flash Player 9.115.0 e versões posteriores e o AIR implementam essa tecnologia (o processo é chamado *mipmapping*), por meio da criação de versões otimizadas de escala variável de cada bitmap (começando em 50%).

O Flash Player 11.3 e o AIR 3.3 suportam o fluxo de texturas pelo parâmetro `streamingLevels` dos métodos `Context3D.createCubeTexture()` e `Context3D.createTexture()`.

A compactação de textura permite que você armazene imagens em formato compactado diretamente na GPU, o que poupa a memória da GPU e a largura de banda da memória. Tipicamente, texturas compactadas são compactadas offline e carregadas na GPU em formato compactado. Entretanto, o Flash Player 11.4 e o AIR 3.4 oferecem suporte à a compactação de textura de tempo de execução, o que é útil em determinadas situações, como a renderização de texturas dinâmicas de arte vetorial. Para usar a compactação de textura de tempo de execução, siga as seguintes etapas:

- Crie o objeto de textura chamando o método `Context3D.createTexture()`, fornecendo `flash.display3D.Context3DTextureFormat.COMPRESSED` ou `flash.display3D.Context3DTextureFormat.COMPRESSED_ALPHA` no terceiro parâmetro.

- Utilizando a instância `flash.display3D.textures.Texture` retornada por `createTexture()`, chame `flash.display3D.textures.Texture.uploadFromBitmapData()` ou `flash.display3D.textures.Texture.uploadFromByteArray()`. Esses métodos carregam e compactam a textura em uma etapa.

Os mapas MIP são criados para os seguintes tipos de bitmaps:

- um bitmap (arquivos JPEG, GIF ou PNG) exibido com a classe Loader do ActionScript 3.0
- um bitmap na biblioteca de um documento Flash Professional
- um objeto BitmapData
- um bitmap exibido com a função `loadMovie()` do ActionScript 2.0

Os mapas MIP não são aplicados a objetos filtrados nem a clipes de filme em cache de bitmap. Entretanto, os mapas MIP são aplicados se você tiver transformações de bitmap em um objeto de exibição filtrado, mesmo se o bitmap estiver dentro de conteúdo mascarado.

Os mapeamentos mip ocorre automaticamente, mas você pode seguir algumas orientações para se certificar de que suas imagens usem essa otimização:

- Para reprodução de vídeo, defina a propriedade `smoothing` como `true` para o objeto Video (consulte a classe Video).
- Para bitmaps, a propriedade `smoothing` não tem que ser definida como `true`, mas os aprimoramentos de qualidade são mais visíveis quando os bitmaps usam suavização.
- Use tamanhos de bitmap que podem ser divididos por 4 ou 8 para imagens de bidimensionais (como 640 x 128, que pode ser reduzido da seguinte forma: 320 x 64 > 160 x 32 > 80 x 16 > 40 x 8 > 20 x 4 > 10 x 2 > 5 x 1).

  Para texturas tridimensionais, use mapas MIP nos quais cada imagem está em uma resolução que é uma potência de 2 (isto é, 2^n). Por exemplo, a imagem principal está em uma resolução de 1024 x 1024 pixels. As imagens de qualidade inferior no mapa MIP teriam 512 x 512, 256 x 256, 128 x 128 até 1 x 1 pixels, totalizando 11 imagens no MIP.

  Observe que o mapeamento mip não ocorre para conteúdo bitmap com largura ou altura ímpar.

Exemplo de bitmap: lua giratória animada

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo de lua giratória animada demonstra técnicas para trabalhar com objetos Bitmap e dados de imagem de bitmap (objetos BitmapData). O exemplo cria uma animação de uma lua giratória e esférica utilizando uma imagem plana da superfície da lua como os dados de imagem não processados. As técnicas a seguir são demonstradas:

- Carregamento de uma imagem externa e acesso aos seus dados de imagem não processados
- Criação de animação por meio de cópia repetida de pixels de diferentes partes de uma imagem de origem
- Criação de uma imagem de bitmap pela definição de valores de pixel

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos de aplicativo da lua giratório animada podem ser encontrados na pasta Samples/SpinningMoon. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
SpinningMoon.mxml ou SpinningMoon.fla	O arquivo principal do aplicativo no Flex (MXML) ou Flash (FLA).
com/example/programmingas3/moon/MoonSphere.as	A classe que executa a funcionalidade de carregamento, exibição e animação da lua.
moonMap.png	O arquivo de imagem que contém uma fotografia da superfície da lua, que é carregado e utilizado para criar a lua giratório e animada.

Carregamento de uma imagem externa como dados de bitmap

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A primeira tarefa principal que essa amostra executa é o carregamento de um arquivo de imagem externa, que é uma fotografia da superfície da lua. A operação de carregamento é manipulada por dois métodos na classe MoonSphere: o construtor `MoonSphere()`, em que o processo de carregamento é iniciado, e o método `imageLoadComplete()`, que é chamado quando a imagem externa está totalmente carregada.

O carregamento de uma imagem externa é semelhante ao carregamento de um SWF externo; ambos utilizam uma ocorrência da classe flash.display.Loader para executar essa operação. O código real no método `MoonSphere()` que inicia o carregamento de uma imagem é o seguinte:

```
var imageLoader:Loader = new Loader();
imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoadComplete);
imageLoader.load(new URLRequest("moonMap.png"));
```

A primeira linha declara que a ocorrência de Loader é chamada `imageLoader`. A terceira linha inicia realmente o processo de carregamento chamando o método `load()` do objeto Loader, transmitindo uma ocorrência de URLRequest que representa a URL da imagem a ser carregada. A segunda linha configura o ouvinte do evento que será acionado quando a imagem for totalmente carregada. Observe que o método `addEventListener()` não é chamado na própria ocorrência de Loader; em vez disso, ele é chamado na propriedade `contentLoaderInfo` do objeto Loader. A própria ocorrência de Loader não despacha eventos relacionados ao conteúdo sendo carregado. A propriedade `contentLoaderInfo`, entretanto, contém uma referência ao objeto LoaderInfo que é associado ao conteúdo sendo carregado no objeto Loader (a imagem externa nesse caso). Aquele objeto LoaderInfo fornece vários eventos relacionados ao progresso e à conclusão do carregamento de conteúdo externo, incluindo o evento `complete` (`Event.COMPLETE`) que acionará uma chamada para o método `imageLoadComplete()` quando a imagem for totalmente carregada.

Embora o início do carregamento da imagem externa seja uma parte importante do processo, é igualmente importante saber o que fazer quando o carregamento for concluído. Como mostrado no código acima, a função `imageLoadComplete()` é chamada quando a imagem é carregada. Aquela função faz várias atividades com os dados de imagem carregados, descritas nas seções subsequentes. Contudo, para usar os dados da imagem, é necessário acessar aqueles dados. Quando o objeto Loader for utilizado para carregar uma imagem externa, a imagem carregada torna-se uma ocorrência de bitmap que é anexada a um objeto de exibição filho do objeto Loader. Nesse caso, a ocorrência de Loader está disponível para o método do ouvinte do evento como parte do objeto de evento que é transmitido para o método como um parâmetro. As primeiras linhas do método `imageLoadComplete()` são como se segue:

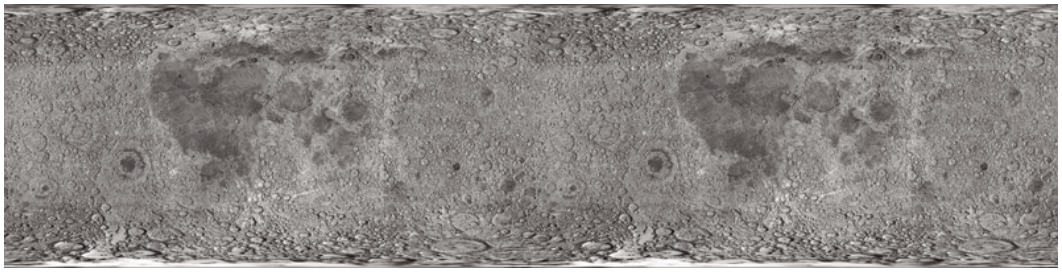
```
private function imageLoadComplete(event:Event):void
{
    textureMap = event.target.content.bitmapData;
    ...
}
```

Observe que o parâmetro do objeto de evento é chamado `event`, e é uma ocorrência da classe Event. Cada ocorrência da classe Event tem uma propriedade `target`, que se refere ao objeto que está acionando o evento (nesse caso, a ocorrência de LoaderInfo na qual o método `addEventListener()` foi chamado, conforme descrito anteriormente). O objeto LoaderInfo, por sua vez, tem uma propriedade `content` que (uma vez concluído o processo de carregamento) contém a ocorrência de Bitmap com a imagem de bitmap carregada. Se você quiser exibir a imagem diretamente na tela, poderá anexar essa ocorrência de Bitmap (`event.target.content`) a um contêiner do objeto de exibição. (Você também pode anexar o objeto Loader a um contêiner do objeto de exibição). Entretanto, nessa amostra, o conteúdo carregado é utilizado como uma origem dos dados da imagem não processados que estão sendo exibidos na tela. Consequentemente, a primeira linha do método `imageLoadComplete()` lê a propriedade `bitmapData` da ocorrência de Bitmap carregada (`event.target.content.bitmapData`) e a armazena na variável de ocorrência chamada `textureMap`, que é usada como uma origem de dados de imagem para criar a animação da lua giratória. Isso é descrito a seguir.

Criação da animação pela cópia de pixels

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma definição básica da animação é a ilusão de movimentação, ou alteração, criada pela alteração de uma imagem ao longo do tempo. Nessa amostra, o objetivo é criar a ilusão de uma lua esférica girando ao redor de seu eixo vertical. Contudo, para o propósito de animação, você pode ignorar o aspecto de distorção esférica da amostra. Considere a imagem real que está carregada e é usada como a origem dos dados da imagem de lua:



Como você pode ver, a imagem não uma nem várias esferas; ela é uma fotografia retangular da superfície da lua. Como a foto foi tirada exatamente na linha do equador da lua, as partes da imagem próximas à parte superior e inferior da imagem são expandidas e distorcidas. Para remover a distorção da imagem e deixá-la com uma aparência esférica, utilizaremos um filtro de mapa de deslocamento, como descrito posteriormente. Entretanto, como essa imagem de origem é um retângulo, para criar a ilusão de que a esfera está girando, o código simplesmente precisa deslizar a foto da superfície da lua horizontalmente.

Observe que a imagem realmente contém duas cópias da fotografia da superfície da lua próximas uma da outra. Essa imagem é a imagem de origem da qual os dados de imagem foram copiados repetidas vezes para criar a aparência de movimento. Tendo duas cópias da imagem próximas uma da outra, um efeito de rolagem contínuo e ininterrupto pode ser criado com mais facilidade. Vamos avançar no processo de animação etapa por etapa para ver como ele funciona.

O processo realmente envolve dois objetos separados do ActionScript. Primeiro, existe a imagem de origem carregada, que no código é representada pela ocorrência de BitmapData chamada `textureMap`. Como descrito anteriormente, `textureMap` é preenchido com os dados de imagem assim que a imagem externa é carregada, utilizando este código:

```
textureMap = event.target.content.bitmapData;
```

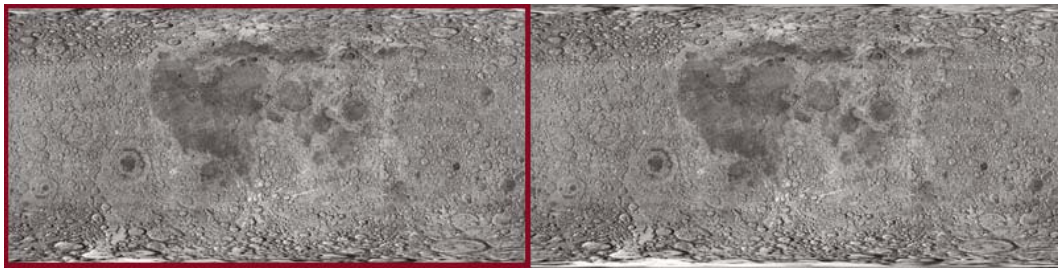
O conteúdo de `textureMap` é a imagem da lua em retângulo. Além disso, para criar a rotação animada, o código usa uma ocorrência de Bitmap chamada `sphere`, que é o objeto de exibição real que mostra a imagem da lua na tela. Como `textureMap`, o objeto `sphere` é criado e preenchido com seus dados de imagem iniciais no método `imageLoadComplete()`, utilizando o seguinte código:

```
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));
```

Como mostra o código, `sphere` é instanciado. Sua propriedade `bitmapData` (os dados de imagem não processados que são exibidos por `sphere`) é criada com a mesma altura e metade da largura de `textureMap`. Em outras palavras, o conteúdo de `sphere` será o tamanho de uma foto da lua (desde que a imagem `textureMap` contenha duas fotos da lua lado a lado). Em seguida, a propriedade `bitmapData` é preenchida com dados de imagem utilizando o método `copyPixels()`. Os parâmetros na chamada do método `copyPixels()` indicam várias coisas:

- O primeiro parâmetro indica que os dados da imagem são copiados de `textureMap`.
- O segundo parâmetro, uma nova ocorrência de Rectangle, especifica de qual parte de `textureMap` o snapshot da imagem deve ser tirado; nesse caso, o snapshot é um retângulo que começa no canto superior esquerdo de `textureMap` (indicado pelos dois primeiros parâmetros `Rectangle(): 0, 0`) e a largura e a altura do snapshot do retângulo corresponde às propriedades `width` e `height` de`sphere`.
- O terceiro parâmetro, uma nova ocorrência de Point com os valores x e y de `0`, define o destino dos dados de pixel — nesse caso, o canto superior esquerdo (0, 0) de `sphere.bitmapData`.

Representado visualmente, o código copia os pixels de `textureMap` contornados na imagem a seguir e os cola em `sphere`. Em outras palavras, o conteúdo de BitmapData de`sphere` é a parte de `textureMap` em destaque aqui:



Lembre-se, contudo, de que esse é apenas o estado inicial de `sphere` — o conteúdo da primeira imagem que é copiada na `sphere`.

Com a imagem de origem carregada e `sphere` criada, a tarefa final executada pelo método `imageLoadComplete()` é a configuração da animação. A animação é orientada pela ocorrência de Timer chamada `rotationTimer`, que é criada e iniciada pelo seguinte código:

```
var rotationTimer:Timer = new Timer(15);
rotationTimer.addEventListener(TimerEvent.TIMER, rotateMoon);
rotationTimer.start();
```

O código cria primeiro a ocorrência de Timer chamada `rotationTimer`; o parâmetro transmitido para o construtor `Timer()` indica que `rotationTimer` deve acionar seu evento `timer` a cada 15 milissegundos. Em seguida, o método `addEventListener()` é chamado, especificando que, quando o evento `timer` (`TimerEvent.TIMER`) ocorre, o método `rotateMoon()` é chamado. Por fim, o temporizado é realmente iniciado chamando seu método `start()`.

Devido ao modo com `rotationTimer` está definido, aproximadamente a cada 15 milissegundos, o Flash Player chama o método `rotateMoon()` na classe MoonSphere, que é onde a animação da lua acontece. O código-fonte do método `rotateMoon()` é o seguinte:

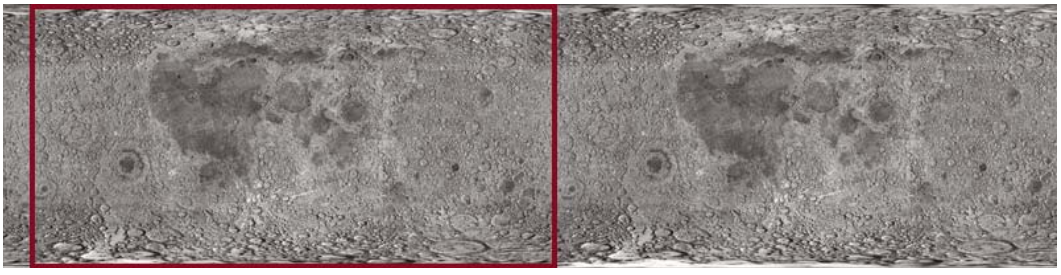
```
private function rotateMoon(event:TimerEvent):void
{
    sourceX += 1;
    if (sourceX > textureMap.width / 2)
    {
        sourceX = 0;
    }

    sphere.Data.copyPixels(textureMap,
                                   new Rectangle(sourceX, 0, sphere.width, sphere.height),
                                   new Point(0, 0));

    event.updateAfterEvent();
}
```

O código faz três coisas:

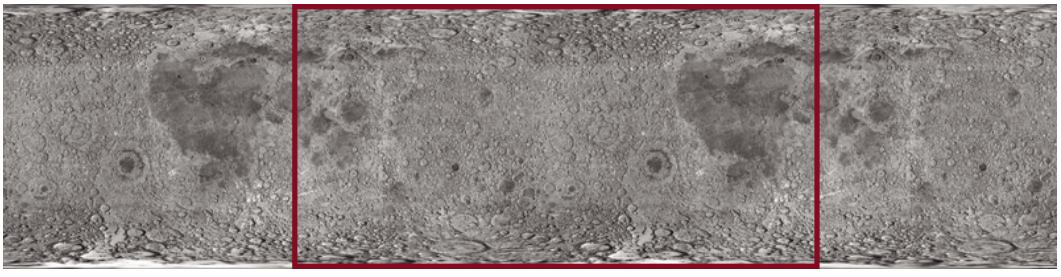
1 O valor da variável `sourceX` (inicialmente definido como 0) é incrementado em 1.

```
sourceX += 1;
```

Como você verá, `sourceX` é usado para determinar o local em `textureMap` do qual os pixels serão copiados para `sphere`; portanto esse código tem o efeito de mover o retângulo um pixel para a direita em `textureMap`. Voltando à representação visual, depois de vários ciclos de animação, o retângulo de origem terá se movido vários pixels para a direita, como se segue:

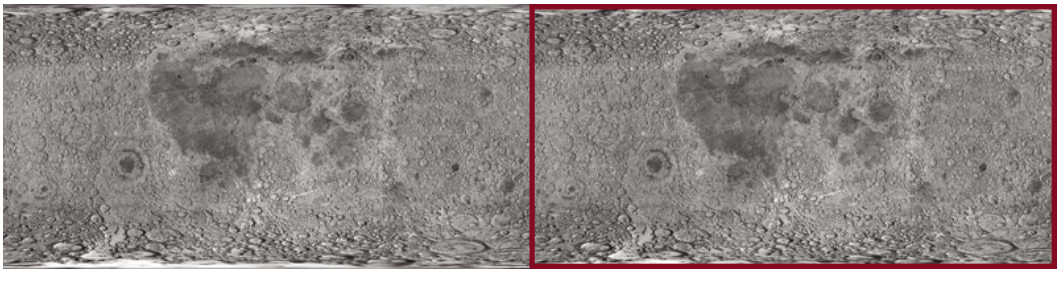


Depois de vários outros ciclos, o retângulo terá se movido para mais longe:



Esse deslocamento gradual, uniforme no local do qual os pixels são copiados é a chave da animação. Movendo lenta e continuamente o local de origem para a direita, a imagem que é exibida na tela em `sphere` parece deslizar continuamente para a esquerda. Essa é a razão pela qual a imagem de origem (`textureMap`) precisa ter duas cópias da foto da superfície da lua. Como o retângulo se move continuamente para a direita, a maioria das vezes ele não está sobre uma única foto da lua, mas sobrepõe as duas fotos da lua.

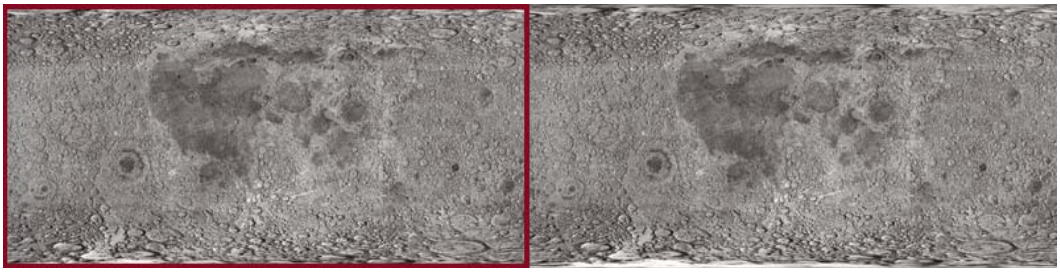
2 Com o retângulo de origem movendo-se lentamente para a direita, há um problema. Finalmente, o retângulo atingirá a borda direita de `textureMap` e ficará sem os pixels da foto da lua para copiar na `sphere`:



As linhas de código a seguir solucionam esse problema:

```
if (sourceX >= textureMap.width / 2)
{
    sourceX = 0;
}
```

O código verifica se `sourceX` (a borda esquerda do retângulo) atingiu o meio de `textureMap`. Em caso afirmativo, ele redefine `sourceX` para 0, movendo-o para a borda esquerda de `textureMap` e iniciando o ciclo novamente:



3 Com o valor `sourceX` apropriado calculado, a etapa final na criação da animação é copiar realmente os pixels do novo retângulo de origem para `sphere`. O código que faz isso é muito semelhante ao código que inicialmente preenche `sphere` (descrito anteriormente); a única diferença é que nesse caso, na chamada do construtor `new Rectangle()`, a borda esquerda do retângulo é colocada em `sourceX`:

```
sphere.bitmapData.copyPixels(textureMap,
                             new Rectangle(sourceX, 0, sphere.width, sphere.height),
                             new Point(0, 0));
```

Lembre-se de que esse código é chamada repetidamente a cada 15 milissegundos. Como o local do retângulo de origem é deslocado de forma contínua, e os pixels são copiados em `sphere`, a aparência na tela é a de que a imagem da foto da lua representada por `sphere` desliza continuamente. Em outras palavras, a lua parece girar continuamente.

Criação da aparência esférica

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A lua, é claro, é uma esfera e não um retângulo. Consequentemente, a mostra precisa pegar a foto da superfície da lua retangular, à medida que é animada continuamente, e convertê-la em uma esfera. Isso envolve duas etapas separadas: uma máscara é usada para ocultar todo o conteúdo exceto uma região circular da foto da superfície da lua, e um filtro de mapa de deslocamento é usado para distorcer a aparência da foto da lua para que ela pareça tridimensional.

Primeiro, uma máscara no formato de círculo é usada para ocultar todo o conteúdo do objeto MoonSphere exceto a esfera criada pelo filtro. O código a seguir cria a máscara como uma ocorrência de Shape e a aplica como a máscara da ocorrência MoonSphere:

```
moonMask = new Shape();
moonMask.graphics.beginFill(0);
moonMask.graphics.drawCircle(0, 0, radius);
this.addChild(moonMask);
this.mask = moonMask;
```

Observe que, como MoonSphere é um objeto de exibição (ele é baseado na classe Sprite), a máscara pode ser aplicada diretamente à ocorrência de MoonSphere usando sua propriedade `mask` herdada.

Ocultar simplesmente partes da foto utilizando uma máscara em forma de círculo não é suficiente para criar um efeito de esfera giratória realista. Devido à forma como a foto da superfície da lua foi tirada, suas dimensões não são proporcionais; as partes da imagem que estão mais próximas à parte superior ou inferior da imagem são mais distorcidas e expandidas em comparação com as partes na linha do equador. Para distorcer a aparência da foto da lua e torná-la tridimensional, usaremos um filtro de mapa de deslocamento.

Um filtro de mapa de deslocamento é um tipo de filtro utilizado para distorcer uma imagem. Nesse caso, a foto da lua será "distorcida" para torná-la mais realista, comprimindo a parte superior e inferior da imagem horizontalmente, enquanto deixa o meio inalterado. Considerando que o filtro funciona em uma parte em formato de quadrado da foto, comprimir a parte superior e a inferior, mas não o meio, transformará o quadrado em um círculo. Um efeito colateral da animação dessa imagem distorcida é que o meio da imagem parece mover-se para mais longe em distância real de pixels do que as áreas próximas à parte superior e inferior, o que cria a ilusão de que o círculo é realmente um objeto tridimensional (uma esfera).

O código a seguir é usado para criar o filtro do mapa de deslocamento chamado `displaceFilter`:

```
var displaceFilter:DisplacementMapFilter;
displaceFilter = new DisplacementMapFilter(fisheyeLens,
                                new Point(radius, 0),
                                BitmapDataChannel.RED,
                                BitmapDataChannel.GREEN,
                                radius, 0);
```

Esse primeiro parâmetro, `fisheyeLens`, é conhecido como a imagem de mapa; nesse caso, um objeto BitmapData que é criado de modo programático. A criação daquela imagem é descrita na seção "Criação de uma imagem de bitmap pela definição de valores de pixel" na página 268. Os outros parâmetros descrevem a posição na imagem filtrada na qual o filtro deve ser aplicado, os canais de cor que serão utilizados para controlar o efeito do deslocamento e até qual extensão eles afetarão o deslocamento. Depois que o filtro de mapa de deslocamento é criado, ele é aplicado a `sphere`, ainda dentro do método `imageLoadComplete()`:

```
sphere.filters = [displaceFilter];
```

A imagem final, com a máscara e o filtro de mapa de deslocamento aplicados, se parece com:

Com cada ciclo de animação da lua giratória, o conteúdo de BitmapData da esfera é sobregravado por um novo snapshot dos dados da imagem de origem. Contudo, o filtro não precisa ser reaplicado sempre. Isso porque o filtro é aplicado à ocorrência de Bitmap (o objeto de exibição) em vez de aos dados do bitmap (as informações em pixels não processadas). Lembre-se, a ocorrência do Bitmap não são os dados reais de bitmap; ela é uma objeto de exibição que mostra os dados de bitmap na tela. Para usar uma analogia, uma ocorrência de Bitmap é como o projeto de slides que é usado para exibir slides fotográficos em uma tela; um objeto BitmapData é como o slide de fotografia real que pode ser apresentando por de um projetor de slide. Um filtro pode ser aplicado diretamente a um objeto BitmapData, que seria comparável a desenhar diretamente no slide fotográfico para alterar a imagem. Um filtro também pode ser aplicado a qualquer objeto de exibição, inclusive uma ocorrência de Bitmap; isso seria como colocar um filtro na frente das lentes do projetor de slides para distorcer a saída mostrada na tela (sem alterar o slide original). Como os dados de bitmap não processados estão acessíveis por meio da propriedade bitmapData da ocorrência de Bitmap, o filtro pode ter sido aplicado diretamente nos dados de bitmap não processados. Entretanto, nesse caso, faz sentido aplicar o filtro ao objeto de exibição Bitmap em vez dos dados de bitmap.

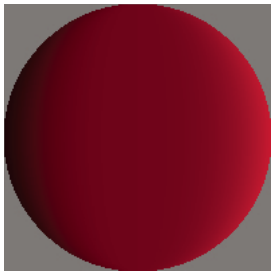
Para obter informações detalhadas sobre o uso de filtro de mapas de deslocamento no ActionScript, consulte "Filtro de objetos de exibição" na página 274.

Criação de uma imagem de bitmap pela definição de valores de pixel

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um importante aspecto de um filtro de mapa de deslocamento é que ele realmente envolve duas imagens. Uma imagem, a imagem de origem, é a imagem que será realmente alterada pelo filtro. Nessa amostra, a imagem de origem é a ocorrência de Bitmap chamada `sphere`. A outra imagem usada pelo filtro é conhecida como a imagem de mapa. A imagem do mapa não é realmente exibida na ela. Em vez disso, a cor de cada um de seus pixels é usada como uma entrada para a função de deslocamento — a cor do pixel em uma determinada coordenada x, y na imagem de mapa determina quanto de deslocamento (deslocamento físico na posição) é aplicado ao pixel naquelas coordenadas x, y na imagem de origem.

Consequentemente, para usar o filtro de mapa de deslocamento para criar um efeito de esfera, a amostra precisa de imagem de mapa apropriada — aquela que tem um plano de fundo cinza e um círculo que é preenchido com um gradiente de uma única cor (vermelho) indo horizontalmente de escuro para claro, como mostrado aqui:



Como somente uma imagem e um filtro de mapa são utilizados nesta amostra, a imagem do mapa é criada apenas uma vez, no método `imageLoadComplete()` (em outras palavras, quando a imagem externa acabar de ser carregada. A imagem do mapa, chamada `fisheyeLens`, é criada chamando o método `createFisheyeMap()` da classe MoonSphere:

```
var fisheyeLens:BitmapData = createFisheyeMap(radius);
```

Dentro do método `createFisheyeMap()`, a imagem de mapa é desenhada um pixel por vez utilizando o método `setPixel()` da classe BitmapData. O código completo para o método `createFisheyeMap()` é listado aqui, seguido por uma discussão etapa por etapa de como ele funciona:

```
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;
}
```

Primeiro, quando o método é chamado, ele recebe um parâmetro, `radius`, indicando o raio da imagem em forma de círculo a ser criado. Em seguida, o código cria o objeto BitmapData no qual o círculo será desenhado. Aquele objeto, chamado `result`, é por fim transmitido de volta como o valor de retorno do método. Como mostrado no snippet de código a seguir, a ocorrência de BitmapData de `result` é criada com a largura e a altura tão grandes quando o diâmetro do círculo, sem transparência (`false` para o terceiro parâmetro), e preenchida com a cor `0x808080` (cinza médio):

```
var result:BitmapData = new BitmapData(diameter,
                                       diameter,
                                       false,
                                       0x808080);
```

Em seguida, o código usa loops para iterar cada pixel da imagem. O loop externo engloba cada coluna da imagem da esquerda para a direita (utilizando a variável `i` para representar a posição horizontal do pixel atualmente sendo manipulado), enquanto o loop interno engloba cada pixel da coluna atual de cima para baixo (com a variável `j` representando a posição vertical de pixels atual). O código para os loops (com o conteúdo do loop interno omitido) é mostrado aqui:

```
for (var i:int = 0; i < diameter; i++)
{
    for (var j:int = 0; j < diameter; j++)
    {
        ...
    }
}
```

À medida que o loop engloba os pixels um a um, em cada pixel um valor (o valor de cor daquele pixel na imagem de mapa) é calculado. Esse processo envolve quatro etapas:

1. O código calcula a distância do pixel atual do centro do círculo ao longo do eixo x (`i - radius`). Esse valor é dividido pelo raio para torná-lo um percentual de raio em vez de uma distância absoluta (`(i - radius) / radius`). Esse valor de percentual é armazenado em uma variável chamada `pctX`, e o valor equivalente para o eixo y é calculado e armazenado na variável `pctY`, como mostra este código:

   ```
   var pctX:Number = (i - radius) / radius;
   var pctY:Number = (j - radius) / radius;
   ```

2 Usando uma fórmula trigonométrica padrão, o Teorema de Pitágoras, a distância linear entre o cento do círculo e o ponto atual é calculado de `pctX` e `pctY`. Esse valor é armazenado em uma variável chamada `pctDistance`, como mostrado aqui:

```
var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);
```

3 Em seguida, o código verifica se o percentual de distância é menor do que 1 (significando 100% do raio, ou em outras palavras, se o pixel sendo considerado está dentro do raio do círculo). Se o pixel estiver dentro do círculo, ele recebe um valor de cor calculado (omitido aqui, mas descrito na etapa 4); em caso negativo, nada mais acontece com aquele pixel; sua cor é deixada como o padrão de cinza médio:

```
if (pctDistance < 1)
{
    ...
}
```

4 Para aqueles pixels que estiverem dentro do círculo, um valor de cor é calculado para o pixel. A cor final será uma sombra de vermelho variando de preto (0% de vermelho) na borda esquerda do círculo até vermelho vivo (100%) na borda direita do círculo. O valor da cor é inicialmente calculado em três partes (vermelho, verde e azul), como mostrado aqui:BitmapData

```
red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
green = 0;
blue = 0;
```

Observe que apenas a parte vermelha da cor (a variável `red`) tem realmente um valor. Os valores de verde e azul (as variáveis `green` e `blue`) são mostrados aqui para fins de explicação, mas podem ser omitidos. Como o propósito desse método é criar um círculo que contenha um gradiente vermelho, nenhum valor de verde ou azul é necessário.

Depois que os três valores de cores individuais estiverem determinados, eles serão combinados em um único valor de cor inteiro utilizando um algoritmo de deslocamento de bits padrão, mostrado neste código:

```
rgb = (red << 16 | green << 8 | blue);
```

Finalmente, com o valor de cor calculado, aquele valor é atribuído ao pixel atual utilizando o método `setPixel()` do objeto `result` BitmapData, mostrado aqui:

```
result.setPixel(i, j, rgb);
```

Decodificação assíncrona de imagens de bitmap

Flash Player 11 e posterior, Adobe AIR 2.6 e posterior

Quando você trabalha com imagens de bitmap, pode decodificar e carregar assincronamente as imagens de bitmap a fim de aprimorar o desempenho percebido de seu aplicativo. A decodificação assíncrona de uma imagem de bitmap, em muitos casos pode durar o mesmo tempo que a decodificação síncrona da imagem. No entanto, a imagem de bitmap é decodificada em um encadeamento separado antes que o objeto `Loader` associado envie o evento `COMPLETE`. Assim, é possível decodificar assincronamente imagens maiores depois de carregá-las.

A classe `ImageDecodingPolicy` no pacote `flash.system` permite especificar o esquema de carregamento de bitmap. O esquema de carregamento padrão é síncrono.

Política de decodificação de bitmap	Esquema de carregamento de bitmap	Descrição
`ImageDecodingPolicy.ON_DEMAND`	Síncrono	As imagens carregadas são decodificadas quando os dados da imagem são acessados. Use esta política para decodificar imagens menores. Você também pode usar esta política quando seu aplicativo não depender de efeitos e transições complexas.
`ImageDecodingPolicy.ON_LOAD`	Assíncrono	As imagens carregadas são decodificadas no carregamento, antes que o evento `COMPLETE` seja despachado. Ideal para imagens maiores (maiores que 10 MP). Quando estiver desenvolvendo aplicativos móveis baseados no AIR com transições de página, use esta política de carregamento de bitmap para aumentar o desempenho percebido de seu aplicativo.

Nota: *Se o arquivo carregado for uma imagem de bitmap e a política de decodificação usada for `ON_LOAD`, a imagem será decodificada de modo assíncrono antes de o evento `COMPLETE`ser despachado.*

O código a seguir mostra o uso da classe `ImageDecodingPolicy`:

```
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);
```

Você ainda pode usar a decodificação ON_DEMAND com os métodos `Loader.load()` e `Loader.loadBytes()`. No entanto, todos os outros métodos que obtêm um objeto `LoaderContext` como argumento ignoram qualquer valor de `ImageDecodingPolicy` passado.

O exemplo a seguir mostra a diferença entre a decodificação síncrona e assíncrona de uma imagem de bitmap:

```
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");
        }
    }
}
```

Para uma demonstração do efeito de diferentes políticas de decodificação, consulte Thibaud Imbert: Decodificação de bitmap assíncrona nos tempos de execução do Adobe Flash

Capítulo 14: Filtro de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Historicamente, o aplicativo de efeitos de filtro às imagens de bitmap são o domínio dos softwares especializados de edição de imagem, como o Adobe Photoshop® e o Adobe Fireworks®. O ActionScript 3.0 inclui o pacote flash.filters, que contém uma série de classes de filtro de efeito de bitmap. Esses efeitos permitem que os desenvolvedores apliquem filtros, em termos programáticos, aos bitmaps e exibam objetos e com isso obtenham muitos dos mesmos efeitos que estão disponíveis nos aplicativos de manipulação gráfica.

Noções básicas sobre filtragem de objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma das formas de acrescentar polonês a um aplicativo é acrescentar efeitos gráficos simples. Você pode acrescentar um efeito de sombra por trás de uma foto para criar a ilusão de 3D, ou um brilho ao redor de um botão para mostrar que ele está ativo. O ActionScript 3.0 inclui dez filtros que podem ser aplicados em qualquer objeto de exibição ou em uma ocorrência de BitmapData. Os filtros incorporados variam do básico, como filtros de sombra e brilho, ao complexo, como o filtro de mapa de mesclagem e o filtro de convolução.

Nota: *Além dos filtros incorporados, você também pode programar filtros e efeitos personalizados usando o Pixel Bender. Consulte "Trabalho com sombreadores Pixel Bender" na página 308.*

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você pode encontrar ao criar filtros:

Bisel Uma borda criada por pixels de iluminação nos dois lados e pixels de escurecimento nos dois lados opostos. Esse efeito cria a aparência de uma borda tridimensional. O efeito é geralmente utilizado para botões erguidos ou recuados e imagens gráficas semelhantes.

Convolução Distorção de pixels em uma imagem por meio da combinação do valor de cada pixel com os valores de alguns ou todos os pixels ao redor, usando diversas proporções.

Deslocamento Mudança ou movimentação de pixels em uma imagem para uma nova posição.

Matriz Uma grade de números usada para realizar alguns cálculos matemáticos aplicando os números da grade em diversos valores e combinando os resultados.

Mais tópicos da Ajuda

Pacote flash.filters

flash.display.DisplayObject.filters

flash.display.BitmapData.applyFilter()

Criação e aplicação de filtros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os filtros permitem aplicar diversos efeitos em objetos de exibição e bitmap, variando desde sombras projetadas até biséis e desfoques. Cada filtro é definido como uma classe, de modo que a aplicação de filtros envolve a criação de ocorrências de objetos de filtro, o que equivale a criar qualquer outro objeto. Depois de criar uma ocorrência de um objeto de filtro, é possível aplicá-la com facilidade em um objeto de exibição usando a propriedade `filters` do objeto ou, no caso de um objeto BitmapData, usando o método `applyFilter()`.

Criação de um filtro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para criar um objeto de filtro, basta chamar o método de construtor da classe de filtro selecionada. Por exemplo, para criar um objeto DropShadowFilter, use o seguinte código:

```
import flash.filters.DropShadowFilter;
var myFilter:DropShadowFilter = new DropShadowFilter();
```

Embora não seja mostrado aqui, o construtor `DropShadowFilter()` (como todos os construtores de classes de filtro) aceita vários parâmetros opcionais que podem ser usados para personalizar a aparência do efeito do filtro.

Aplicação de um filtro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Depois de criar um objeto de filtro, você pode aplicá-lo em um objeto de exibição ou em um objeto BitmapData; o modo de aplicação do filtro depende do objeto no qual ele será aplicado.

Aplicação de um filtro em um objeto de exibição

Use a propriedade `filters` para aplicar efeitos de filtro em um objeto de exibição. A propriedade `filters` de um objeto de exibição é uma ocorrência de Array, cujos elementos são objetos de filtro aplicados no objeto de exibição. Para aplicar um único filtro em um objeto de exibição, crie a ocorrência de filtro, adicione-a a uma ocorrência de Array e atribua esse objeto Array à propriedade `filters` do objeto de exibição:

```
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;
```

Se desejar atribuir vários filtros ao objeto, basta adicionar todos os filtros à ocorrência de Array antes de atribuí-la à propriedade `filters`. Para adicionar vários objetos a uma matriz, transmita-os como parâmetros para o construtor. Por exemplo, esse código aplica um filtro de bisel e um filtro de brilho ao objeto de exibição criado anteriormente:

```
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;
```

Ao criar a matriz que contém os filtros, você pode usar o construtor `new Array()` (conforme mostrado nos exemplos anteriores) ou a sintaxe do literal de matriz, colocando os filtros entre colchetes (`[]`). Por exemplo, esta linha de código:

```
var filters:Array = new Array(dropShadow, blur);
```

faz a mesma coisa que esta linha de código:

```
var filters:Array = [dropShadow, blur];
```

Se vários filtros forem aplicados em objetos de exibição, a aplicação será feita de modo cumulativo e sequencial. Por exemplo, se uma matriz de filtros tiver dois elementos, um filtro de bisel adicionado primeiro e um filtro de sombra projetada adicionado depois, o filtro de sombra projetada será aplicado no filtro de bisel e no objeto de exibição. Isso ocorre devido à posição do filtro de sombra projetada em segundo lugar na matriz de filtros. Se desejar aplicar filtros de modo não cumulativo, você deve aplicar cada filtro em uma nova cópia do objeto de exibição.

Se estiver atribuindo apenas um ou alguns filtros a um objeto de exibição, crie a ocorrência do filtro e atribua-a ao objeto em uma única instrução. Por exemplo, a linha de código a seguir aplica um filtro de desfoque em um objeto de exibição chamado `myDisplayObject`:

```
myDisplayObject.filters = [new BlurFilter()];
```

O código anterior cria uma ocorrência de Array usando a sintaxe do literal de matriz (entre colchetes), cria uma ocorrência de BlurFilter como um elemento da matriz e atribui a essa matriz a propriedade `filters` do objeto de exibição chamado `myDisplayObject`.

Remoção de filtros de um objeto de exibição

A remoção de todos os filtros de um objeto de exibição é tão simples quanto atribuir um valor nulo à propriedade `filters`:

```
myDisplayObject.filters = null;
```

Se tiver aplicado vários filtros em um objeto e desejar remover apenas um dos filtros, realize as etapas para alterar a matriz da propriedade `filters`. Para obter mais informações, consulte "Possíveis problemas resultantes do trabalho com filtros" na página 277.

Aplicação de um filtro em um objeto BitmapData

A aplicação de um filtro em um objeto BitmapData requer o uso do método `applyFilter()` do objeto BitmapData:

```
var rect:Rectangle = new Rectangle();
var origin:Point = new Point();
myBitmapData.applyFilter(sourceBitmapData, rect, origin, new BlurFilter());
```

O método `applyFilter()` aplica um filtro em um objeto BitmapData de origem, produzindo uma nova imagem filtrada. Esse método não modifica a imagem de origem inicial; em vez disso, o resultado da aplicação do filtro na imagem de origem é armazenado na ocorrência de BitmapData na qual o método `applyFilter()` é chamado.

Como funcionam os filtros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A filtragem de objetos de exibição funciona armazenando uma cópia do objeto original em cache como um bitmap transparente.

Assim que um filtro é aplicado em um objeto de exibição, o tempo de execução armazena o objeto em cache como um bitmap enquanto o objeto tiver uma lista de filtros válida. O bitmap de origem é usado como imagem original para todos os efeitos de filtro aplicados posteriormente.

Cada objeto de exibição geralmente contém dois bitmaps: um com o objeto original de exibição de origem e outro para a imagem final, após a filtragem. A imagem final é usada no momento da renderização. Desde que o objeto de exibição não seja alterado, a imagem final não precisa de atualização.

Possíveis problemas resultantes do trabalho com filtros

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Existem várias fontes de possíveis problemas que devem ser consideradas ao trabalhar com filtros.

Cache de bitmaps e filtros

Para aplicar um filtro em um objeto de exibição, o cache de bitmaps deve ser ativado para esse objeto. Quando você aplica um filtro a um objeto de exibição cuja propriedade `cacheAsBitmap` está definida como `false`, a propriedade `cacheAsBitmap` do objeto é automaticamente definida como `true`. Se mais tarde você remover todos os filtros do objeto de exibição, a propriedade `cacheAsBitmap` é definida para o último valor em que estava.

Alteração de filtros em tempo de execução

Se um objeto de exibição já tiver um ou mais filtros aplicados, não será possível alterar o conjunto de filtros adicionando mais filtros ou removendo filtros da matriz da propriedade `filters`. Em vez disso, para adicionar ou alterar o conjunto de filtros aplicados, faça as alterações em uma matriz separada e, em seguida, atribua essa matriz à propriedade filters do objeto de exibição para que os filters sejam aplicados no objeto. A maneira mais simples de fazer isso é ler a matriz da propriedade `filters` em uma variável Array e fazer as modificações nessa matriz temporária. Depois, atribua essa matriz novamente à propriedade `filters` do objeto de exibição. Em casos mais complexos, talvez seja necessário manter uma matriz de filtros principal separada. Faça as alterações nessa matriz de filtros principal e atribua-a novamente à propriedade `filters` do objeto de exibição depois de cada alteração.

Adição de mais um filtro

O código a seguir demonstra o processo de adição de mais um filtro a um objeto de exibição que já tem um ou mais filtros aplicados. Inicialmente, um filtro de brilho é aplicado no objeto de exibição chamado `myDisplayObject`; posteriormente, quando o objeto de exibição é clicado, a função `addFilters()` é chamada. Nesta função, dois filtros adicionais são aplicados em `myDisplayObject`:

```
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);
```

Remoção de um filtro do conjunto de filtros

Se um objeto de exibição tiver vários filtros aplicados e você desejar remover um dos filtros enquanto os outros continuam a ser aplicados no objeto, copie os filtros em uma matriz separada, remova o filtro indesejado dessa matriz e atribua a matriz temporária novamente à propriedade `filters` do objeto de exibição. Várias maneiras de remover um ou mais elementos de qualquer matriz são descritas em "Recuperação de valores e remoção de elementos de matriz" na página 32.

O mais fácil é remover o filtro de nível superior do objeto (o último filtro aplicado no objeto). Use o método `pop()` da classe Array para remover o filtro da matriz:

```
// 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;
```

Similarmente, para remover o filtro de nível inferior (o primeiro aplicado no objeto), use o mesmo código, usando o método `shift()` da matriz Array em vez do método `pop()`.

Para remover um filtro do meio de uma matriz de filtros (supondo que a matriz tem mais de dois filtros), use o método `splice()`. Você deve saber o índice (a posição na matriz) do filtro que deseja remover. Por exemplo, o código a seguir remove o segundo filtro (no índice 1) de um objeto de exibição:

```
// 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;
```

Determinação do índice de um filtro

Você precisa saber qual filtro deve ser removido da matriz para determinar o índice do filtro. Você deve saber (em virtude da designação do aplicativo) ou calcular o índice do filtro a ser removido.

A melhor maneira é designar o aplicativo de modo que o filtro a ser removido sempre fique na mesma posição no conjunto de filtros. Por exemplo, se houver apenas um objeto de exibição com um filtro de convolução e um filtro de sombra projetada aplicados (nessa ordem) e você desejar remover o filtro de sombra projetada, mas manter o outro, o filtro estará em uma posição conhecida (o filtro de nível superior) para que você saiba com antecedência qual método Array deve ser usado (neste caso, `Array.pop()` para remover o filtro de sombra projetada).

Se o filtro que deseja remover for sempre do mesmo tipo, mas não estiver necessariamente na mesma posição do conjunto de filtros todas as vezes, verifique o tipo de dados de cada filtro da matriz para determinar qual deve ser removido. Por exemplo, o código a seguir determina qual é o filtro de brilho em um conjunto de filtros e remove esse filtro do conjunto.

```
// 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;
```

Em um caso mais complexo, como se o filtro a ser removido for selecionado em tempo de execução, o melhor é manter uma cópia separada permanente da matriz de filtros que serve como a lista principal de filtros. Sempre que você alterar o conjunto de filtros, altere a lista principal e aplique essa matriz de filtros como a propriedade `filters` do objeto de exibição.

Por exemplo, na listagem de código a seguir, vários filtros de convolução são aplicados em um objeto de exibição para criar efeitos visuais diferentes e, posteriormente, um desses filtros é removido enquanto os outros são mantidos. Neste caso, o código mantém uma cópia principal da matriz de filtros, bem como uma referência ao filtro a ser removido. Localizar e remover o filtro específico é similar à abordagem anterior, mas, em vez de criar uma cópia temporária da matriz de filtros, a cópia principal é manipulada e aplicada no objeto de exibição.

```
// 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;
}
```

Nesta abordagem (quando estiver comparando uma referência de filtro armazenada com os itens da matriz de filtros para determinar qual filtro deve ser removido), você *deve* manter uma cópia separada da matriz de filtros - o código não funcionará se a referência de filtro armazenada for comparada com os elementos de uma matriz temporária copiada da propriedade `filters` do objeto de exibição. Isso ocorre porque internamente, quando uma matriz é atribuída à propriedade `filters`, o tempo de execução faz uma cópia de cada objeto de filtro na matriz. Essas cópias (não os objetos originais) são aplicadas no objeto de exibição e, quando a propriedade `filters` é lida em uma matriz temporária, a matriz temporária contém referências aos objetos de filtro copiados, em vez de referências aos objetos de filtro originais. Consequentemente, se no exemplo anterior você tentar determinar o índice de `filterToRemove` comparando-o com os filtros de uma matriz temporária, nenhuma correspondência será encontrada.

Transformações de objeto e filtros

Nenhuma região filtrada (uma sombra projetada, por exemplo) fora do retângulo delimitador do objeto de exibição é considerada como parte da superfície para fins de detecção de ocorrências (determinar se uma ocorrência se sobrepõe ou faz interseção com outra). Como os métodos de detecção de ocorrências da classe DisplayObject são baseados em vetores, não é possível detectar ocorrências no resultado de bitmap. Por exemplo, se um filtro de bisel for aplicado em uma ocorrência de botão, a detecção de ocorrências não estará disponível na parte do bisel da ocorrência.

Os filtros não permitem dimensionar, girar e inclinar objetos; se o objeto de exibição filtrado for dimensionado (se `scaleX` e `scaleY` não forem 100%), o efeito de filtro não será dimensionado com a ocorrência. Isso significa que a forma original da ocorrência é girada, dimensionada ou inclinada; no entanto, o filtro não é girado, dimensionado ou inclinado com a ocorrência.

Você pode animar uma ocorrência com um filtro para criar efeitos realistas ou aninhar ocorrências e usar a classe BitmapData para animar os filtros e atingir esse efeito.

Objetos de bitmap e filtros

Quando algum filtro é aplicado em um objeto BitmapData, a propriedade `cacheAsBitmap` é definida automaticamente como `true`. Desse modo, o filtro é realmente aplicado na cópia do objeto, não no original.

Em seguida, essa cópia é colocada na exibição principal (no objeto original) o mais perto possível do pixel mais próximo. Se os limites do bitmap original forem alterados, o bitmap de cópia filtrado será recriado a partir do original, em vez de ser esticado ou distorcido.

Se você apagar todos os filtros de um objeto de exibição, a propriedade `cacheAsBitmap` será redefinida como o valor original antes do filtro ser aplicado.

Filtros de exibição disponíveis

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 inclui dez classes de filtro que podem ser aplicadas em objetos de exibição e em objetos BitmapData:

- Filtro de bisel (classe BevelFilter)
- Filtro de desfoque (classe BlurFilter)
- Filtro de sombra projetada (classe DropShadowFilter)
- Filtro de brilho (classe GlowFilter)
- Filtro de bisel de gradiente (classe GradientBevelFilter)
- Filtro de brilho de gradiente (classe GradientGlowFilter)
- Filtro de matriz de cor (classe ColorMatrixFilter)
- Filtro de convolução (classe ConvolutionFilter)
- Filtro de mapa de deslocamento (classe DisplacementMapFilter)
- Filtro de sombreador (classe ShaderFilter)

Os seis primeiros filtros são simples e podem ser usados para criar um efeito específico, com alguns recursos de personalização disponíveis. Esses seis filtros podem ser aplicados usando o ActionScript e também a objetos no Flash Professional usando o painel Filtros. Consequentemente, mesmo que os filtros sejam aplicados com ActionScript, se você tiver o Flash Professional, poderá usar a interface visual para experimentar rapidamente filtros e configurações diferentes para saber como criar o efeito desejado.

Os quatro últimos filtros estão disponíveis somente no ActionScript. Esses filtros (matriz de cor, convolução, mapa de deslocamento e sombreador) são muito mais flexíveis nos tipos de efeitos que podem criar. Em vez de serem otimizados para um único efeito, eles fornecem poder e flexibilidade. Por exemplo, selecionando valores diferentes para sua matriz, o filtro de convolução pode ser usado para criar efeitos como desfoque, entalhe, nitidez, localização de bordas de cor, transformações e muito mais.

Cada filtro, simples ou complexo, pode ser personalizado com suas propriedades. Geralmente, existem duas opções para configurar propriedades de filtro. Todos os filtros permitem definir as propriedades por meio da transmissão de valores de parâmetro ao construtor do objeto de filtro. Se preferir, independentemente de configurar as propriedades de filtro transmitindo parâmetros, você pode ajustar os filtros posteriormente definindo valores para as propriedades do objeto de filtro. A maioria das listagens de código de exemplo define as propriedades diretamente para facilitar o acompanhamento do exemplo. Entretanto, você normalmente atinge o mesmo resultado em menos linhas de código passando os valores como parâmetros no construtor de objeto de filtro. Para obter mais detalhes sobre os princípios de cada filtro, suas propriedades e seus parâmetros de construção, consulte Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Filtro de bisel

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe BevelFilter permite a adição de uma borda 3D chanfrada ao objeto filtrado. Esse filtro deixa os cantos ou bordas pontudos do objeto talhados ou chanfrados.

As propriedades da classe BevelFilter permitem personalizar a aparência do bisel. É possível definir cores de realce e sombra, desfoques de borda chanfrada, ângulos de bisel e colocação da borda chanfrada; você pode inclusive criar um efeito vazado.

O exemplo a seguir carrega uma imagem externa e aplica um filtro de bisel nela.

```
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];
```

Filtro de desfoque

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe BlurFilter mancha ou desfoca objetos de exibição e seu respectivo conteúdo. Os efeitos de desfoque são úteis para dar a impressão de que um objeto está fora de foco ou para simular um deslocamento rápido, como em um desfoque de movimento. Defina a propriedade `quality` do filtro de desfoque como muito baixa para simular um efeito de lente ligeiramente fora do foco. Definir a propriedade `quality` como alta resulta em um efeito de desfoque suave similar a um desfoque de Gauss.

O exemplo a seguir cria um objeto circular usando o método `drawCircle()` da classe Graphics e aplica um filtro de desfoque nele:

```
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];
```

Filtro de sombra projetada

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As sombras projetadas dão a impressão de que existe uma fonte de luz separada situada acima do objeto de destino. A posição e a intensidade dessa fonte de luz podem ser modificadas para produzir diversos efeitos de sombra projetada diferentes.

A classe DropShadowFilter usa um algoritmo similar ao do filtro de desfoque. A principal diferença é que o filtro de sombra projetada tem mais propriedades que podem ser modificadas para simular outros atributos de fonte de luz (como alfa, cor, deslocamento e brilho).

O filtro de sombra projetada também permite aplicar opções de transformação personalizadas no estilo da sombra projetada, incluindo a sombra interna ou externa e o modo vazado (também conhecido como recorte).

O código a seguir cria uma caixa quadrada e aplica um filtro de sombra projetada nela:

```
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];
```

Filtro de brilho

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe GlowFilter aplica um efeito de iluminação aos objetos de exibição, como se o objeto fosse iluminado por baixo, criando um leve brilho.

Assim como o filtro de sombra projetada, o filtro de brilho inclui propriedades para modificar a distância, o ângulo e a cor da fonte de luz para produzir efeitos variados. O GlowFilter também tem várias opções para modificar o estilo do brilho, incluindo o brilho interno ou externo e o modo vazado.

O código a seguir cria uma transversal usando a classe Sprite e aplica um filtro de brilho nela:

```
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];
```

Filtro de bisel de gradiente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe GradientBevelFilter permite a aplicação de um efeito de bisel aprimorado em objetos de exibição ou objetos BitmapData. Usar uma cor de gradiente no bisel melhora muito a profundidade espacial do bisel, dando às bordas uma aparência 3D mais realista.

O código a seguir cria um objeto retangular usando o método `drawRect()` da classe Shape e aplica um filtro de bisel de gradiente nele.

```
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);
```

Filtro de brilho de gradiente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe GradientGlowFilter permite a aplicação de um efeito de brilho aprimorado em objetos de exibição ou objetos BitmapData. Os efeitos permitem que você tenha mais controle sobre o brilho e produza um efeito mais realista. Além disso, o filtro de brilho de gradiente permite aplicar um brilho gradiente nas bordas interna, externa ou superior de um objeto.

O exemplo a seguir desenha um círculo no palco e aplica um filtro de brilho de gradiente a ele. À medida que você move o mouse mais para a direita e para baixo, a quantidade de desfoque aumenta nas direções horizontal e vertical, respectivamente. Além disso, sempre que você clicar no palco, a intensidade do desfoque aumentará.

```
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);
```

Exemplo: combinação de filtros básicos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O código a seguir usa vários filtros básicos, combinados com um cronômetro, para criar ações repetidas e simular um farol animado.

```
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();
```

Filtro de matriz de cor

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe ColorMatrixFilter é usada para manipular a cor e os valores alfa do objeto filtrado. Isso permite criar alterações de saturação, rotação de matizes (alterando uma paleta de uma gama de cores para outra), alterações de luminância para alfa e outros efeitos de manipulação de cor usando valores de um canal de cor e possivelmente aplicando-os em outros canais.

Conceitualmente, o filtro percorre cada pixel da imagem de origem e separa-os nos componentes vermelho, verde, azul e alfa. Em seguida, os valores fornecidos na matriz de cor são multiplicados por cada um desses valores, somando os resultados para determinar o valor de cor resultante que será exibido na tela para o pixel em questão. A propriedade `matrix` do filtro é uma matriz de 20 números usada no cálculo da cor final. Para obter detalhes do algoritmo específico utilizado para calcular os valores de cor, consulte a entrada que descreve a propriedade `matrix`da classe ColorMatrixFilter em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Filtro de convolução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe ConvolutionFilter pode ser usada para aplicar diversas transformações de imagem em objetos BitmapData e objetos de exibição, como desfoque, detecção de borda, nitidez, entalhe e bisel.

Conceitualmente, o filtro de convolução percorre cada pixel da imagem de origem e determina a cor final desse pixel usando o valor do pixel e dos pixels ao redor. Uma matriz, especificada como uma matriz de valores numéricos, indica até que ponto o valor de cada pixel ao redor afeta o valor resultante final.

Considere o tipo de matriz utilizado com mais frequência, que é uma matriz 3 x 3. A matriz inclui nove valores:

```
N   N   N
N   P   N
N   N   N
```

Quando o filtro de convolução é aplicado a um certo pixel, é considerado o valor de cor do pixel propriamente dito ("P" no exemplo), bem como os valores dos pixels ao redor ("N" no exemplo). No entanto, definindo valores na matriz, você especifica a prioridade de determinados pixels na imagem resultante.

Por exemplo, a matriz a seguir, que tem um filtro de convolução aplicado, deixará uma imagem exatamente como era:

```
0   0   0
0   1   0
0   0   0
```

A imagem não é alterada porque o valor do pixel original tem uma intensidade relativa igual a 1 para determinar a cor final do pixel, enquanto os valores dos pixels ao redor têm a intensidade relativa igual a 0 - suas cores não afetam a imagem final.

Similarmente, essa matriz fará com que os pixels de uma imagem mudem um pixel para a esquerda:

```
0   0   0
0   0   1
0   0   0
```

Observe que, nesse caso, o pixel propriamente dito não afeta o valor final do pixel exibido nesse local na imagem final; apenas o valor do pixel à direita é usado para determinar o valor resultante do pixel.

No ActionScript, você cria a matriz como uma combinação de uma ocorrência de Array que contém os valores e as duas propriedades que especificam o número de linhas e colunas da matriz. O exemplo a seguir carrega uma imagem e, quando o carregamento termina, aplica um filtro de convolução na imagem usando a matriz da listagem anterior:

```
// 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);
```

O que não fica óbvio nesse código é o efeito do uso de valores diferentes de 1 ou 0 na matriz. Por exemplo, a mesma matriz, com o número 8 em vez de 1 na posição direita, executa a mesma ação (movimentando os pixels para a esquerda). Além disso, ela afeta as cores da imagem, deixando-as oito vezes mais brilhantes. Isso ocorre porque os valores finais de cor do pixel são calculados pela multiplicação dos valores da matriz pelas cores do pixel original, pela soma dos valores e pela divisão pelo valor da propriedade `divisor` do filtro. Observe que, no código de exemplo, a propriedade `divisor` é definida como 1. Como regra geral, se desejar que o brilho das cores permaneça quase igual ao brilho da imagem original, o divisor deve ser igual à soma dos valores da matriz. Desse modo, se uma matriz tiver a soma de seus valores igual a 8 e o divisor igual a 1, a imagem resultante será aproximadamente 8 vezes mais brilhante do que a imagem original.

Embora o efeito nessa matriz não seja muito notável, outros valores de matriz podem ser usados para criar diversos efeitos. Veja alguns conjuntos padrão de valores de matriz para diferentes efeitos usando uma matriz 3 x 3:

- Desfoque básico (divisor 5):

```
 0 1 0
 1 1 1
 0 1 0
```

- Nitidez (divisor 1):

```
 0, -1, 0
-1, 5, -1
 0, -1, 0
```

- Detecção de borda (divisor 1):

```
 0, -1, 0
-1, 4, -1
 0, -1, 0
```

- Efeito de entalhe (divisor 1):

```
-2, -1, 0
-1, 1, 1
 0, 1, 2
```

  Observe que na maioria desses efeitos o divisor é igual a 1. Isso ocorre porque a adição de valores de matriz negativos a valores de matriz positivos resulta em 1 (ou 0 no caso da detecção de borda, mas o valor da propriedade `divisor` não pode ser 0).

Filtro de mapa de deslocamento

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe DisplacementMapFilter usa valores de pixel de um objeto BitmapData (conhecido como a imagem do mapa de deslocamento) para realizar um efeito de deslocamento em um novo objeto. A imagem do mapa de deslocamento normalmente é diferente do objeto de exibição ou da ocorrência de BitmapData real no qual o filtro está sendo aplicado. Um efeito de deslocamento envolve a movimentação de pixels nas imagem filtrada - em outras palavras, deslocá-los do local original para alguma posição. Esse filtro pode ser usado para criar um efeito alterado, distorcido ou matizado.

O local e a quantidade de deslocamento aplicados em um determinado pixel são especificados pelo valor de cor da imagem do mapa de deslocamento. Ao trabalhar com o filtro, além de especificar a imagem do mapa, você especifica os seguintes valores para controlar como o deslocamento é calculado a partir da imagem do mapa:

- Ponto do mapa: o local da imagem filtrada na qual o canto superior esquerdo do filtra de deslocamento será aplicado. Use esse ponto apenas se desejar aplicar o filtro em parte de uma imagem.
- Componente X: canal de cor da imagem do mapa que afeta a posição x dos pixels.
- Componente Y: canal de cor da imagem do mapa que afeta a posição y dos pixels.
- Escala X: um valor de multiplicador que especifica a intensidade do deslocamento do eixo x.
- Escala Y: um valor de multiplicador que especifica a intensidade do deslocamento do eixo y.
- Modo de filtro: determina o que fazer em qualquer espaço vazio criado pelos pixels que estão sendo alterados. As opções, definidas como constantes na classe DisplacementMapFilterMode, devem exibir os pixels originais (modo de filtro `IGNORE`), colocar os pixels em torno do outro lado da imagem (modo de filtro `WRAP`, que é o padrão), usar o pixel alterado mais próximo (modo de filtro `CLAMP`) ou preencher os espaços com uma cor (modo de filtro `COLOR`).

Para entender o funcionamento do filtro do mapa de deslocamento, considere um exemplo simples. No código a seguir, uma imagem é carregada e, em seguida, centralizada no palco. Um filtro de mapa de deslocamento é aplicado, fazendo com que os pixels da imagem inteira mudem horizontalmente para a esquerda.

```
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);
```

As propriedades usadas para definir o deslocamento são as seguintes:

- Bitmap de mapa: o bitmap de deslocamento é uma nova ocorrência de BitmapData criada pelo código. Suas dimensões coincidem com as dimensões da imagem carregada (de modo que o deslocamento é aplicado na imagem inteira). Ela é preenchida com pixels vermelhos sólidos.
- Ponto do mapa: esse valor é definido como o ponto 0, 0 - mais uma vez, o deslocamento será aplicado na imagem inteira.
- Componente X: esse valor é definido como a constante `BitmapDataChannel.RED`, o que significa que o valor vermelho do bitmap de mapa determinará o deslocamento dos pixels (o quanto se movimentam) ao longo do eixo x.
- Escala X: esse valor é definido como 250. O valor total de deslocamento (a partir da imagem do mapa que está completamente vermelha) desloca a imagem apenas um pouco (aproximadamente metade de um pixel), de modo que, se esse valor fosse definido como 1, a imagem seria movida apenas 0,5 pixel na horizontal. Se esse valor for definido como 250, a imagem será deslocada aproximadamente 125 pixels.

Essas configurações fazem com que os pixels da imagem filtrada sejam deslocados 250 pixels para a esquerda. A direção (esquerda ou direita) e o valor do deslocamento baseiam-se no valor de cor dos pixels da imagem do mapa. Conceitualmente, o filtro percorre cada pixel da imagem filtrada (pelo menos os pixels da região onde o filtro é aplicado que, nesse caso, são todos os pixels) e faz o seguinte com cada pixel:

1. Localiza o pixel correspondente na imagem do mapa. Por exemplo, quando o filtro calcula o valor de deslocamento para o pixel no canto superior esquerdo da imagem filtrada, será observado o pixel no canto superior esquerdo da imagem do mapa.
2. Determina o valor do canal de cor especificado no pixel do mapa. Nesse caso, o canal de cor do componente x é o canal vermelho, de modo que o filtro observa qual é o valor do canal vermelho da imagem do mapa no pixel em questão. Como a imagem do mapa é vermelho sólido, o canal vermelho do pixel é 0xFF ou 255. Esse valor é usado como valor de deslocamento.
3. Compara o valor de deslocamento com o valor "central" (127, que é o valor médio entre 0 e 255). Se o valor de deslocamento for inferior ao valor médio, o pixel será deslocado em uma direção positiva (para a direita no deslocamento x; para baixo no deslocamento y). Por outro lado, se o valor de deslocamento for maior do que o valor médio (como neste exemplo), o pixel será deslocado em uma direção negativa (para a esquerda no deslocamento x; para cima no deslocamento y). Para ser mais preciso, o filtro subtrai o valor de deslocamento de 127 e o resultado (positivo ou negativo) é o valor relativo do deslocamento que é aplicado.
4. Finalmente, ele determina o valor real de deslocamento definindo qual porcentagem de deslocamento completo é representado pelo valor de deslocamento relativo. Nesse caso, vermelho total indica 100% de deslocamento. Essa porcentagem é multiplicada pelo valor da escala x ou y para determinar o número de pixels do deslocamento que será aplicado. Neste exemplo, 100% vezes um multiplicador igual a 250 determina o valor de deslocamento - aproximadamente 125 pixels para a esquerda.

Como nenhum valor foi especificado para o componente e a escala y, os padrões (que não provocam nenhum deslocamento) foram usados; é por esse motivo que a imagem não é alterada na vertical.

A configuração padrão do modo de filtro, `WRAP`, é usada no exemplo, de modo que os pixels se movem para a esquerda e o espaço vazio à direita é preenchido pelos pixels que se deslocaram da margem esquerda da imagem. Você pode experimentar esse valor para ver os diferentes efeitos. Por exemplo, se você adicionar a linha a seguir à parte do código onde as propriedades de deslocamento estão sendo definidas (antes da linha `loader.filters = [displacementMap]`), a imagem parecerá ter sido manchada ao longo do palco:

```
displacementMap.mode = DisplacementMapFilterMode.CLAMP;
```

Em exemplos mais complexos, a listagem a seguir usa um filtro de mapa de deslocamento para criar um efeito de lupa em uma imagem:

```
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);
```

Primeiro, o código gera dois círculos de gradiente, que são combinados para formar a imagem do mapa de deslocamento. O círculo vermelho cria o deslocamento do eixo x (`xyFilter.componentX = BitmapDataChannel.RED`) e o círculo azul cria o deslocamento do eixo y (`xyFilter.componentY = BitmapDataChannel.BLUE`). Para ajudar você a entender como se parece a imagem do mapa de deslocamento, o código adiciona os círculos originais, bem como o círculo combinado que serve como a imagem do mapa, à parte inferior da tela.

Em seguida, o código carrega uma imagem e, à medida que o mouse se move, aplica o filtro de deslocamento na parte da imagem que está embaixo do mouse. Os círculos de gradiente usados como a imagem do mapa de deslocamento fazem com que a região deslocada se distancie do ponteiro. Observe que as regiões de cinza da imagem do mapa não provocam nenhum deslocamento. A cor de cinza é `0x7F7F7F`. Os canais azul e vermelho dessa sombra de cinza correspondem exatamente à sombra central desses canais de cor, de modo que não há nenhum deslocamento em uma área de cinza da imagem do mapa. Do mesmo modo, não há nenhum deslocamento no centro do círculo. Embora a cor não seja cinza, os canais azul e vermelho dessa cor são idênticos aos canais azul e vermelho do cinza médio e, como azul e vermelho são as cores que provocam o deslocamento, não ocorre nenhum deslocamento aqui.

Filtro de sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A classe ShaderFilter permite o uso de um efeito de filtro personalizado definido como um sombreador Pixel Bender. Como é gravado como um sombreador Pixel Bender, o efeito de filtro pode ser completamente personalizado. O conteúdo filtrado é transmitido para o sombreador como uma entrada de imagem e o resultado da operação de shader se transforma no resultado de filtro.

Nota: *O filtro Sombreador está disponível no ActionScript a partir do Flash Player 10 e do Adobe AIR 1.5.*

Para aplicar um filtro de sombreador em um objeto, crie uma ocorrência de Shader que representa o sombreador Pixel Bender utilizado. Para obter informações sobre o procedimento de criação de uma ocorrência de Shader e sobre como especificar a imagem de entrada e valores de parâmetro, consulte "Trabalho com sombreadores Pixel Bender" na página 308.

Ao usar um sombreador como filtro, tenha em mente três coisas importantes:

- O sombreador deve ser definido para aceitar pelo menos uma imagem de entrada.

- O objeto filtrado (o objeto de exibição ou o objeto BitmapData no qual o filtro é aplicado) é transmitido para o sombreador como o primeiro valor da imagem de entrada. Devido a isso, não especifique manualmente um valor para a primeira entrada de imagem.
- Se o sombreador definir mais de uma imagem de entrada, as entradas adicionais deverão ser especificadas manualmente (isto é, será necessário definir a propriedade `input` de todas as ocorrências de ShaderInput que pertencem à ocorrência de Shader).

Assim que um objeto Shader tiver sido definido para o sombreador, crie uma ocorrência de ShaderFilter. Este é o objeto de filtro real usado como qualquer outro filtro. Para criar um ShaderFilter que usa um objeto Shader, chame o construtor `ShaderFilter()` e transmita o objeto Shader como um argumento, como mostra esta listagem:

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
```

Para obter um exemplo completo de como utilizar um filtro de sombreador, consulte "Uso de um sombreador como filtro" na página 326.

Exemplo de filtragem de objetos de exibição: bancada do filtro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Filter Workbench fornece uma interface de usuário para aplicar filtros diferentes em imagens e outros conteúdos visuais e para visualizar o código resultante que pode ser usado para gerar o mesmo efeito no ActionScript. Além de fornecer uma ferramenta para experimentar filtros, esse aplicativo demonstra as seguintes técnicas:

- Criação de ocorrências de vários filtros
- Aplicação de vários filtros em um objeto de exibição

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Filter Workbench estão localizados na pasta Amostras/FilterWorkbench. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
com/example/programmingas3/filterWorkbench/FilterWorkbenchController.as	Classe que fornece a principal funcionalidade do aplicativo, incluindo a alternância do conteúdo para os filtros aplicados e a aplicação de filtros no conteúdo.
com/example/programmingas3/filterWorkbench/IFilterFactory.as	Interface que define métodos comuns que são implementados por cada classe padrão de filtro. Essa interface define a funcionalidade comum que a classe FilterWorkbenchController usa para interagir com classes individuais de filtro padrão.
Na pasta com/example/programmingas3/filterWorkbench/: BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	Conjunto de classes, cada uma delas implementa a interface IFilterFactory. Cada uma dessas classes permite criar e definir valores para um único tipo de filtro. Os painéis de propriedade de filtro do aplicativo usam essas classes padrão para criar ocorrências de filtros específicos, que a classe FilterWorkbenchController recupera e aplica no conteúdo da imagem.
com/example/programmingas3/filterWorkbench/IFilterPanel.as	Interface que define métodos comuns implementados pelas classes que definem os painéis da interface de usuário usados para manipular valores de filtro no aplicativo.
com/example/programmingas3/filterWorkbench/ColorStringFormatter.as	Classe de utilitário que inclui um método para converter um valor de cor numérico em um formato de string hexadecimal
com/example/programmingas3/filterWorkbench/GradientColor.as	Classe que serve como objeto de valor, combinando em um único objeto os três valores (cor, alfa e proporção) associados a cada cor em GradientBevelFilter e GradientGlowFilter
Interface de usuário (Flex)	
FilterWorkbench.mxml	O arquivo principal que define a interface de usuário do aplicativo.
flexapp/FilterWorkbench.as	Classe que gera a funcionalidade da interface de usuário do aplicativo principal; essa classe é usada como a classe code-behind do arquivo MXML do aplicativo.
Na pasta flexapp/filterPanels: BevelPanel.mxml BlurPanel.mxml ColorMatrixPanel.mxml ConvolutionPanel.mxml DropShadowPanel.mxml GlowPanel.mxml GradientBevelPanel.mxml GradientGlowPanel.mxml	Conjunto de componentes MXML que gera a funcionalidade de cada painel usado para definir opções para um único filtro.

Arquivo	Descrição
flexapp/ImageContainer.as	Um objeto de exibição que serve como contêiner da imagem carregada na tela
flexapp/controls/BGColorCellRenderer.as	Renderizador de célula personalizado usado para alterar a cor de fundo de uma célula no componente DataGrid
flexapp/controls/QualityComboBox.as	Controle personalizado que define uma caixa combinada que pode ser usada na configuração de qualidade de diversos painéis de filtro.
flexapp/controls/TypeComboBox.as	Controle personalizado que define uma caixa combinada que pode ser usada na configuração de tipo de diversos painéis de filtro.
Interface de usuário (Flash)	
FilterWorkbench.fla	O arquivo principal que define a interface de usuário do aplicativo.
flashapp/FilterWorkbench.as	Classe que gera a funcionalidade da interface de usuário do aplicativo principal; essa classe é usada como a classe document do arquivo FLA do aplicativo.
Na pasta flashapp/filterPanels: BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	Conjunto de classes que geram a funcionalidade de cada painel usado para definir opções para um único filtro. Para cada classe, também existe um símbolo MovieClip associado na biblioteca do arquivo FLA do aplicativo principal, cujo nome corresponde ao nome da classe (por exemplo, o símbolo “BlurPanel” está vinculado à classe definida em BlurPanel.as). Os componentes que fazem parte da interface de usuário são posicionados e nomeados nesses símbolos.
flashapp/ImageContainer.as	Um objeto de exibição que serve como contêiner da imagem carregada na tela
flashapp/BGColorCellRenderer.as	Renderizador de célula personalizado usado para alterar a cor de fundo de uma célula no componente DataGrid
flashapp/ButtonCellRenderer.as	Renderizador de célula personalizado usado para incluir um componente de botão em uma célula no componente DataGrid
Conteúdo da imagem filtrada	
com/example/programmingas3/filterWorkbench/ImageType.as	Essa classe serve como um objeto de valor que contém o tipo e a URL de um único arquivo de imagem no qual o aplicativo pode carregar e aplicar filtros. A classe também inclui um conjunto de constantes que representam os arquivos de imagem reais disponíveis.
images/sampleAnimation.swf, images/sampleImage1.jpg, images/sampleImage2.jpg	Imagens e outros conteúdos visuais nos quais os filtros são aplicados no aplicativo.

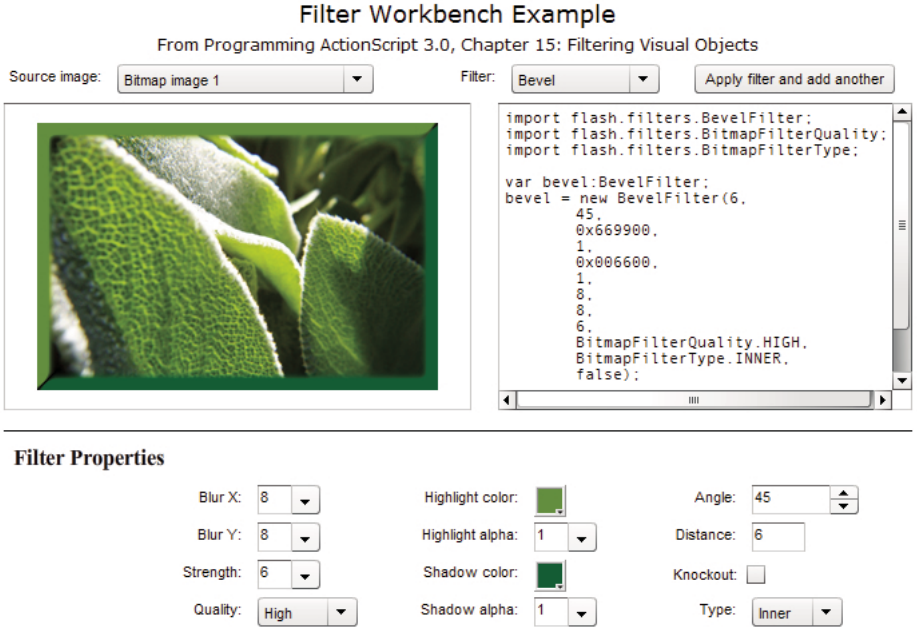
Como experimentar filtros do ActionScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo Filter Workbench foi desenvolvido para ajudar você a experimentar os diversos efeitos de filtro e gerar o código do ActionScript relevante para esse efeito. O aplicativo permite selecionar entre três arquivos diferentes que possuem conteúdo visual, incluindo imagens de bitmap e uma animação criada no Flash, além de permitir aplicar oito filtros diferentes do ActionScript na imagem selecionada, individualmente ou em combinação com outros filtros. O aplicativo inclui estes filtros:

- Bisel (flash.filters.BevelFilter)
- Desfoque (flash.filters.BlurFilter)
- Matrix de cor (flash.filters.ColorMatrixFilter)
- Convolução (flash.filters.ConvolutionFilter)
- Sombra projetada (flash.filters.DropShadowFilter)
- Brilho (flash.filters.GlowFilter)
- Bisel de gradiente (flash.filters.GradientBevelFilter)
- Brilho de gradiente (flash.filters.GradientGlowFilter)

Depois que o usuário seleciona uma imagem e um filtro a ser aplicado nessa imagem, o aplicativo exibe um painel com controles para definir as propriedades específicas do filtro selecionado. Por exemplo, a imagem a seguir mostra o aplicativo com o filtro de bisel selecionado:



À medida que o usuário ajusta as propriedades de filtro, a visualização é atualizada em tempo real. O usuário também pode aplicar vários filtros personalizando um filtro, clicando no botão Apply filter and add another, personalizando outro filtro, clicando no botão Apply filter and add another e assim por diante.

Existem alguns recursos e limitações nos painéis de filtro do aplicativo:

- O filtro de matriz de cor inclui um conjunto de controles para manipular diretamente propriedades comuns de imagem, como brilho, contrastes, saturação e matiz. Além disso, é possível especificar valores personalizados de matriz de cor.
- O filtro de convolução, que está disponível somente no ActionScript, inclui no conjunto de valores de matriz de convolução usados normalmente ou valores personalizados podem ser especificados. No entanto, enquanto a classe ConvolutionFilter aceita qualquer tamanho de matriz, o aplicativo Filter Workbench usa uma matriz 3 x 3 fixa, o tamanho de filtro usado com mais frequência.
- Os filtros de mapa de deslocamento e de sombreador, disponíveis somente no ActionScript, não estão presentes no aplicativo Filter Workbench.

Criação de ocorrências de filtro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo Filter Workbench inclui um conjunto de classes, uma para cada filtro disponível, que são usadas por painéis individuais para criar os filtros. Quando o usuário seleciona um filtro, o código do ActionScript associado ao painel de filtro cria uma ocorrência da classe de filtro de fábrica apropriada. Essas classes são conhecidas como *classes de fábrica* porque sua finalidade é criar ocorrências de outros objetos, assim como uma fábrica real cria produtos individuais.

Sempre que o usuário altera um valor de propriedade no painel, o código do painel chama o método adequado na classe de fábrica. Cada classe de fábrica inclui métodos específicos usados pelo painel para criar a ocorrência de filtro adequada. Por exemplo, se o usuário selecionar o filtro de desfoque, o aplicativo criará uma ocorrência de BlurFactory. A classe BlurFactory inclui um método `modifyFilter()` que aceita três parâmetros: `blurX`, `blurY` e `quality`, que são usados em conjunto para criar a ocorrência de BlurFilter desejada:

```
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));
}
```

Por outro lado, se o usuário selecionar o filtro de convolução, esse filtro dará uma flexibilidade muito maior e, consequentemente, terá um conjunto maior de propriedades para controlar. Na classe ConvolutionFactory, o código a seguir é chamado quando o usuário seleciona um valor diferente no painel de filtro:

```
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));
}
```

Observe que, em cada caso, quando os valores de filtro são alterados, o objeto de fábrica envia um evento `Event.CHANGE` para notificar os ouvintes que os valores do filtro foram alterados. A classe FilterWorkbenchController, que realmente aplica os filtros no conteúdo filtrado, ouve esse evento para recuperar uma nova cópia do filtro e reaplicá-lo no conteúdo filtrado quando necessário.

A classe FilterWorkbenchController não precisa conhecer os detalhes específicos de cada classe de fábrica de filtro - ela só precisa saber que o filtro foi alterado e acessar uma cópia do filtro. Para dar suporte a isso, o aplicativo inclui uma interface, IFilterFactory, que define o comportamento que uma classe de fábrica de filtro precisa fornecer para que a ocorrência de FilterWorkbenchController do aplicativo possa fazer seu trabalho. A classe IFilterFactory define o método `getFilter`() que é usado na classe FilterWorkbenchController:

```
function getFilter():BitmapFilter;
```

Observe que a definição do método da interface `getFilter()` especifica o retorno de uma ocorrência de BitmapFilter, em vez de um tipo específico de filtro. A classe BitmapFilter não define um tipo específico de filtro. Em vez disso, BitmapFilter é a classe básica a partir da qual todas as classes de filtro são criadas. Cada classe de fábrica de filtro define uma implementação específica do método `getFilter()` que retorna uma referência ao objeto de filtro criado. Por exemplo, veja uma versão abreviada do código-fonte da classe ConvolutionFactory:

```
public class ConvolutionFactory extends EventDispatcher implements IFilterFactory
{
    // ------- Private vars -------
    private var _filter:ConvolutionFilter;
    ...
    // ------- IFilterFactory implementation -------
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}
```

Na implementação da classe ConvolutionFactory do método `getFilter()`, é retornada uma ocorrência de ConvolutionFilter, embora nenhum objeto que chame `getFilter()` saiba necessariamente disso - de acordo com a definição do método `getFilter()` seguida por ConvolutionFactory, qualquer ocorrência de BitmapFilter deve ser retornada, a qual pode ser uma ocorrência de qualquer classe de filtro do ActionScript.

Aplicação de filtros em objetos de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Conforme explicado antes, o aplicativo Filter Workbench usa uma ocorrência da classe FilterWorkbenchController (a partir daqui mencionada como "ocorrência do controlador"), que realmente aplica os filtros no objeto visual selecionado. Antes de aplicar um filtro, a ocorrência do controlador precisa saber em qual imagem ou conteúdo visual o filtro deve ser aplicado. Quando o usuário seleciona uma imagem, o aplicativo chama o método `setFilterTarget()` na classe FilterWorkbenchController, transmitindo uma das constantes definidas na classe ImageType:

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, targetLoadComplete);
    ...
}
```

Usando essas informações, a ocorrência do controlador carrega o arquivo designado, armazenando-o em uma variável da ocorrência chamada `_currentTarget` assim que o carregamento termina:

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

Quando o usuário seleciona um filtro, o aplicativo chama o método `setFilter()` da ocorrência do controlador, dando ao controlador uma referência ao objeto de fábrica de filtro relevante, armazenado em uma variável da ocorrência chamada `_filterFactory`.

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

Observe que, conforme descrito anteriormente, a ocorrência do controlador não sabe o tipo de dados específico da ocorrência de fábrica de filtro fornecida; ela só sabe que o objeto implementa a ocorrência de IFilterFactory, indicando que tem um método `getFilter()` e envia um evento `change` (`Event.CHANGE`) quando o filtro é alterado.

Quando o usuário altera as propriedades no painel do filtro, a ocorrência do controlador descobre que o filtro foi alterado por meio do evento `change` da fábrica de filtro, que chama o método `filterChange()` da ocorrência do controlador. Esse método, por sua vez, chama o método `applyTemporaryFilter()`:

```
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();
}
```

A aplicação do filtro no objeto de exibição ocorre no método `applyTemporaryFilter()`. Primeiro, o controlador recupera uma referência ao objeto de filtro chamando o método `getFilter()` da fábrica de filtro.

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

A ocorrência do controlador tem uma variável da ocorrência de Array chamada `_currentFilters`, na qual são armazenados todos os filtros que foram aplicados no objeto de exibição. A próxima etapa é adicionar o filtro recém atualizado a essa matriz:

```
_currentFilters.push(currentFilter);
```

Em seguida, o código atribui a matriz dos filtros à propriedade `filters` do objeto de exibição, que realmente aplica os filtros na imagem:

```
_currentTarget.filters = _currentFilters;
```

Finalmente, como esse filtro adicionado recentemente ainda é o filtro de “trabalho”, ele não deve ser aplicado permanentemente no objeto de exibição e, por isso, é removido da matriz `_currentFilters`:

```
_currentFilters.pop();
```

A remoção desse filtro da matriz não afeta o objeto de exibição filtrado porque um objeto de exibição cria uma cópia da matriz de filtros quando esta é atribuída à propriedade `filters` e usa essa matriz interna em vez da original. Desse modo, todas as alterações feitas na matriz dos filtros não afetam o objeto de exibição até que a matriz seja novamente atribuída à propriedade `filters` do objeto de exibição.

Capítulo 15: Trabalho com sombreadores Pixel Bender

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O Adobe Pixel Bender Toolkit permite que os desenvolvedores gravem sombreadores para criar efeitos gráficos e executar outro processamento de imagem e dados. O código de bytes do Pixel Bender pode ser executado no ActionScript para aplicar o efeito aos dados da imagem ou ao conteúdo visual. O uso de sombreadores Pixel Bender no ActionScript fornece a capacidade de criar efeitos visuais personalizados e de executar processamento de dados além dos recursos internos do ActionScript.

Nota: *O suporte para Pixel Bender foi disponibilizado a partir do Flash Player 10 e do Adobe AIR 1.5. Mesclagens, filtros e preenchimentos do Pixel Bender não contam com suporte quando a renderização é feita pela GPU.*

Mais tópicos da Ajuda

Centro de tecnologia do Adobe Pixel Bender

Guia do desenvolvedor do Pixel Bender

Referência do Pixel Bender

flash.display.Shader

flash.filters.ShaderFilter

Introdução ao Pixel Bender para Flash

Introdução ao Pixel Bender para Flex

Noções básicas de sombreadores Pixel Bender

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O Adobe Pixel Bender é uma linguagem de programação utilizada para criar ou manipular conteúdo de imagem. Usando o Pixel Bender, você criar um kernel, também conhecido como sombreador. O sombreador define uma única função executada individualmente em cada pixel de uma imagem. O resultado de cada chamada à função é a cor gerada nessa coordenada de pixel da imagem. É possível especificar valores de imagens e parâmetros de entrada para personalizar a operação. Em uma única execução de um sombreador, os valores de entrada e parâmetro são constantes. A única coisa variável é a coordenada do pixel cuja cor é o resultado da chamada à função.

Quando possível, a função de sombreador é chamada para várias coordenadas de pixel de saída em paralelo. Isso melhora o desempenho do sombreador e pode oferecer processamento de alto desempenho.

No ActionScript, três tipos de efeitos podem ser facilmente criados com um sombreador:

- preenchimento de desenho
- modo de mesclagem
- filtro

Também é possível executar um sombreador no modo autônomo. No modo autônomo, o resultado de um sombreador é acessado diretamente, sem a necessidade de especificar previamente seu uso pretendido. É possível acessar o resultado como dados de imagem ou como dados binários ou numéricos. Não é necessário que os dados sejam dados de imagem. Dessa forma, você pode atribuir a um sombreador um conjunto de dados como entrada. O sombreador processa os dados, e você pode acessar o resultado retornado por ele.

O suporte a Pixel Bender está disponível para o Flash Player 10 e o Adobe AIR 1.5. As combinações, filtros e preenchimentos do Pixel Bender não são suportados na renderização da GPU. Em dispositivos móveis, os sombreadores Pixel Bender são executados com renderização pela CPU. No entanto, o desempenho não é do mesmo nível de um computador desktop. Muitos programas sombreadores podem executar somente alguns quadros por segundo.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você vai encontrar ao criar e usar os sombreadores do Pixel Bender:

Kernel No Pixel Bender, um kernel é o mesmo que um sombreador. Usando o Pixel Bender, seu código define um kernel, que por sua vez define uma única função que é executada individualmente em cada pixel de uma imagem.

Código de bytes do Pixel Bender Um kernel do Pixel Bender é transformado em código de bytes do Pixel Bender. O código de bytes é acessado e executado no tempo de execução.

Linguagem Pixel Bender A linguagem de programação usada para criar um kernel do Pixel Bender.

Toolkit do Pixel Bender O aplicativo usado para criar um arquivo de código de bytes do Pixel Bender com base no código-fonte do Pixel Bender. O kit de ferramentas permite programar, testar e compilar o código-fonte do Pixel Bender.

Shader Neste documento, um sombreador é um conjunto de funcionalidades criado na linguagem Pixel Bender. O código de um sombreador cria um efeito visual ou executa um cálculo. Em qualquer um dos casos, o sombreador retorna um conjunto de dados (normalmente, os pixels de uma imagem). O sombreador realiza a mesma operação em cada ponto de dados, com a única diferença sendo as coordenadas do pixel de saída. O sombreador não é escrito em ActionScript. Ele é criado na linguagem Pixel Bender e compilado no código de bytes do Pixel Bender. Ele pode ser incorporado a um arquivo SWF em tempo de compilação ou carregado como arquivo externo em tempo de execução. Nos dois casos, ele é acessado no ActionScript através da criação de um objeto Shader e da vinculação deste ao código de bytes do sombreador.

Entrada do sombreador Uma entrada complexa, geralmente dados de imagem de bitmap, fornecida para um sombreador para uso em seus cálculos. Para cada variável de entrada definida em um sombreador, é usado um único valor (isto é, uma única imagem ou um conjunto de dados binários) para a execução inteira do sombreador.

Parâmetro de sombreador Um único valor (ou conjunto limitado de valores) que é fornecido para um sombreador usá-lo em seus cálculos. Cada valor de parâmetro é definido para uma única execução do sombreador, e o mesmo valor é usado durante toda a execução.

Teste por meio dos exemplos de código

Talvez você queira testar as listagens do código de exemplo que são fornecidas. Testar o código envolve executá-lo e ver os resultados no SWF que é criado. Todos os exemplos criam conteúdo usando a API de desenho que utiliza ou é modificada pelo efeito de sombreador.

A maioria das listagens de código de exemplo têm duas partes. Uma parte é o código-fonte do Pixel Bender referente ao sombreador usado no exemplo. Primeiro você deve usar o Pixel Bender Toolkit para compilar o código-fonte em um arquivo de código de bytes do Pixel Bender. Siga estas etapas para criar o arquivo de código de bytes do Pixel Bender:

1. Abra o Adobe Pixel Bender Toolkit. Se necessário, no menu Criar, escolha "Ativar avisos e erros do Flash Player".
2. Copie a listagem de código do Pixel Bender e cole-a no painel do editor de código do Pixel Bender Toolkit.
3. No menu Arquivo, escolha "Exportar filtro de kernel do Flash Player".
4. Salve o arquivo do código de bytes do Pixel Bender no mesmo diretório que o documento do Flash. O nome do arquivo deve ser igual ao nome especificado na descrição do exemplo.

A parte do ActionScript de cada exemplo é escrita como arquivo de classe. Execute as etapas a seguir par testar o exemplo no Flash Professional:

1. Crie um documento do Flash vazio e salve-o no seu computador.
2. Crie e salve um novo arquivo do ActionScript no mesmo diretório do documento Flash. O nome do arquivo deve corresponder ao nome da classe na listagem de código. Por exemplo, se a listagem de código define uma classe chamada MyApplication, use o nome MyApplication.as para salvar o arquivo do ActionScript.
3. Copie a listagem de código no arquivo do ActionScript e salve o arquivo.
4. No documento Flash, clique em uma parte em branco do Palco ou espaço de trabalho para ativar o Inspetor de propriedades do documento.
5. No Inspetor de propriedades, no campo Classe do documento, digite o nome da classe ActionScript que você copiou do texto.
6. Execute o programa usando Controlar > Testar filme

   Você verá os resultados do exemplo na janela de visualização.

As técnicas para testar listas de código de exemplo são explicadas em mais detalhes em "Como Usar Exemplos do ActionScript" na página 1139.

Carregamento ou incorporação de um sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A primeira etapa do uso de um sombreador Pixel Bender no ActionScript é obter acesso ao sombreador no seu código ActionScript. Como um sombreador é criado usando o Adobe Pixel Bender Toolkit e gravado na linguagem Pixel Bender, ele não pode ser acessado diretamente no ActionScript. Em vez disso, crie uma ocorrência da classe Shader que represente o sombreador Pixel Bender para o ActionScript. O objeto Shader permite localizar informações sobre o sombreador, como por exemplo, se ele espera parâmetros ou valores da imagem de entrada. Você passa o objeto Shader para outros objetos para realmente usar o sombreador. Por exemplo, para usar o sombreador como um filtro, você atribui o objeto Shader à propriedade `shader` de um objeto ShaderFilter. Como alternativa, para usar o sombreador como um preenchimento de desenho, você passa o objeto Shader como um argumento para o método `Graphics.beginShaderFill()`.

Seu código ActionScript pode acessar um sombreador criado pelo Adobe Pixel Bender Toolkit (um arquivo .pbj) de duas maneiras:

- Carregado em tempo de execução: o arquivo sombreador pode ser carregado como um ativo externo usando um objeto URLLoader. Essa técnica é semelhante ao carregamento de um ativo externo, como um arquivo de texto. O exemplo a seguir demonstra o carregamento de um arquivo de código de bytes do sombreador em tempo de execução e sua vinculação com uma ocorrência de Shader:

```
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
}
```

- Incorporado no arquivo SWF: o arquivo sombreador pode ser incorporado no arquivo SWF em tempo de compilação usando a tag de metadados `[Embed]`. A tag de metadados `[Embed]` fica disponível somente se você usar o SDK do Flex para compilar o arquivo SWF. O parâmetro `source` da tag `[Embed]` aponta para o arquivo sombreador, e seu parâmetro `mimeType` é o `"application/octet-stream"`, como neste exemplo:

```
[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
```

Nos dois casos, você vincula o código de bytes brutos do sombreador (a propriedade `URLLoader.data` ou uma ocorrência da classe de dados `[Embed]`) com a ocorrência de Shader. Como demonstram os exemplos anteriores, você pode atribuir o código de bytes à ocorrência de Shader de duas maneiras. Você pode passar o código de bytes do sombreador como um argumento para o construtor `Shader()`. Você também pode defini-lo como a propriedade `byteCode` da ocorrência de Shader.

Depois que um sombreador Pixel Bender foi criado e vinculado com um objeto Shader, você pode usá-lo para criar efeitos de várias maneiras. Pode usá-lo como um filtro, um modo de mesclagem, um preenchimento bitmap ou para processamento autônomo de bitmap ou de outros dados. Você também pode usar a propriedade `data` do objeto Shader para acessar os metadados de sombreador, especificar imagens de entrada e definir valores de parâmetros.

Acesso aos metadados de sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Ao criar um kernel de sombreador Pixel Bender, o autor pode especificar metadados sobre o sombreador no código fonte Pixel Bender. Ao usar um sombreador no ActionScript, você pode examiná-lo e extrair seus metadados.

Ao criar uma ocorrência de Shader e vinculá-la a um sombreador Pixel Bender, um objeto ShaderData que contém dados sobre o sombreador é criado e armazenado na propriedade `data` do objeto Shader. A classe ShaderData não define nenhuma propriedade própria. No entanto, em tempo de execução uma propriedade é adicionada dinamicamente ao objeto ShaderData para cada valor de metadados definido no código fonte do sombreador. O nome fornecido a cada propriedade é igual ao nome especificado nos metadados. Por exemplo, suponha que o código fonte de um sombreador Pixel Bender inclua a seguinte definição de metadados:

```
namespace : "Adobe::Example";
vendor : "Bob Jones";
version : 1;
description : "Creates a version of the specified image with the specified brightness.";
```

O objeto ShaderData criado para esse sombreador é criado com as seguintes propriedades e valores:

- `namespace` (String): `"Adobe::Example"`
- `vendor` (String): `"Bob Jones"`
- `version` (String): `"1"`
- `description` (String): `"Cria uma versão da imagem especificada com o brilho especificado"`

Como as propriedades de metadados são adicionadas dinamicamente ao objeto ShaderData, você pode usar um loop `for..in` para examinar o objeto ShaderData. Usando essa técnica é possível descobrir se o sombreador tem algum metadado e quais são os seus valores. Além das propriedades de metadados, um objeto ShaderData pode ter propriedades que representam entradas e parâmetros definidos no sombreador. Ao usar um loop `for..in` para examinar um objeto ShaderData, verifique o tipo de dados de cada propriedade para determinar se a propriedade é uma entrada (uma ocorrência de ShaderInput), um parâmetro (uma ocorrência de ShaderParameter) ou um valor de metadados (uma ocorrência de String). O exemplo a seguir mostra como usar um loop `for..in` para examinar as propriedades dinâmicas da propriedade `data` de um sombreador. Cada valor de metadados é adicionado a uma ocorrência de Vector denominada `metadata`. Observe que este exemplo pressupõe que uma ocorrência de Shader denominada `myShader` já foi criada:

```
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
```

Para obter uma versão desse exemplo que também extrai entradas e parâmetros de sombreador, consulte “Identificação de entradas e parâmetros de sombreador” na página 313. Para obter mais informações sobre propriedades de entrada e parâmetro, consulte “Especificação de valores de entrada e de parâmetro de sombreador” na página 313.

Especificação de valores de entrada e de parâmetro de sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Muitos sombreadores Pixel Bender são definidos para usar uma ou mais imagens de entrada que são usadas no processamento do sombreador. Por exemplo, é comum um sombreador aceitar uma imagem de origem e produzi-la com um efeito específico aplicado a ela. Dependendo de como o sombreador é usado, o valor de entrada pode ser especificado automaticamente ou você pode precisar fornecer um valor explicitamente. De modo semelhante, muitos sombreadores especificam parâmetros que são usados para personalizar a saída do sombreador. Você também deve definir explicitamente um valor para cada parâmetro antes de usar o sombreador.

Você usa a propriedade `data` do objeto Shader para definir entradas e parâmetros do sombreador e para determinar se um sombreador específico espera entradas ou parâmetros. A propriedade `data` é uma ocorrência de ShaderData.

Identificação de entradas e parâmetros de sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A primeira etapa da especificação de valores de entrada e de parâmetro de sombreador é descobrir se o sombreador específico que você está usando espera qualquer imagem ou parâmetro de entrada. Cada ocorrência de Shader tem uma propriedade `data` que contém um objeto ShaderData. Se o sombreador definir quaisquer entradas ou parâmetros, eles serão acessados como propriedades desse objeto ShaderData. Os nomes de propriedades correspondem aos nomes especificados para as entradas e parâmetros no código fonte do sombreador. Por exemplo, se um sombreador definir uma entrada denominada `src`, o objeto ShaderData terá uma propriedade denominada `src` que representa essa entrada. Cada propriedade que representa uma entrada é uma ocorrência de ShaderInput, e cada propriedade que representa um parâmetro é uma ocorrência de ShaderParameter.

O ideal é que o autor do sombreador forneça documentação do sombreador, indicando quais valores de imagens de entrada e de parâmetros o sombreador espera, o que eles representam, os valores apropriados, e assim por diante.

No entanto, se o sombreador não estiver documentado (e você não tiver seu código fonte), você poderá inspecionar os dados do sombreador para identificar as entradas e parâmetros. As propriedades que representam entradas e parâmetros são adicionadas dinamicamente ao objeto ShaderData. Consequentemente, você pode usar um loop `for..in` para examinar o objeto ShaderData para descobrir se o sombreador associado define quaisquer entradas ou parâmetros. Conforme descrito em "Acesso aos metadados de sombreador" na página 312, qualquer valor de metadados definido para um sombreador também é acessado como uma propriedade dinâmica adicionada à propriedade `Shader.data`. Ao usar essa técnica para identificar entradas e parâmetros de sombreador, verifique o tipo de dados das propriedades dinâmicas. Se uma propriedade for uma ocorrência de ShaderInput, ela representará uma entrada. Se ela for uma ocorrência de ShaderParameter, ela representará um parâmetro. Caso contrário, ela será um valor de metadados. O exemplo a seguir mostra como usar um loop `for..in` para examinar as propriedades dinâmicas da propriedade `data` de um sombreador. Cada entrada (objeto ShaderInput) é adicionada a uma ocorrência de Vector denominada `inputs`. Cada parâmetro (objeto ShaderParameter) é adicionado a uma ocorrência de Vector denominada `parameters`. Finalmente, quaisquer propriedades de metadados são adicionadas a uma ocorrência de Vector denominada `metadata`. Observe que este exemplo pressupõe que uma ocorrência de Shader denominada `myShader` já foi criada:

```
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
```

Especificação de valores de entrada de sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Muitos sombreadores esperam uma ou mais imagens de entrada que são usadas no processamento do sombreador. No entanto, em muitos casos, uma entrada é especificada automaticamente quando o objeto Shader é usado. Por exemplo, suponha que um sombreador exija uma entrada e seja usado como um filtro. Quando o filtro é aplicado a um objeto de exibição ou a um objeto BitmapData, esse objeto é definido automaticamente como a entrada. Nesse caso você não define explicitamente um valor de entrada.

No entanto, em alguns casos, especialmente se um sombreador definir várias entradas, você definirá explicitamente um valor para uma entrada. Cada entrada definida em um sombreador é representada no ActionScript por um objeto ShaderInput. O objeto ShaderInput é uma propriedade da ocorrência de ShaderData na propriedade `data` do objeto Shader, conforme descrito em "Identificação de entradas e parâmetros de sombreador" na página 313. Por exemplo, suponha que um sombreador defina uma entrada denominada `src` e que o sombreador esteja vinculado a um objeto Shader denominado `myShader`. Nesse caso, você acessa o objeto ShaderInput que corresponde à entrada `src` usando o seguinte identificador:

```
myShader.data.src
```

Cada objeto ShaderInput tem uma propriedade `input` usada para definir o valor da entrada. Você define a propriedade `input` como uma ocorrência de BitmapData para especificar dados da imagem. Também pode definir a propriedade `input` como uma ocorrência de BitmapData ou Vector.<Number> para especificar dados binários ou numéricos. Para obter detalhes e restrições sobre o uso de uma instância de BitmapData ou Vector.<Number> como entrada, consulte a lista `ShaderInput.input` em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Além da propriedade `input`, um objeto ShaderInput tem propriedades que podem ser usadas para determinar qual tipo de imagem a entrada espera. Essas propriedades incluem as propriedades `width`, `height` e `channels`. Cada objeto ShaderInput também tem uma propriedade `index` que é útil para determinar se um valor explícito deve ser fornecido para a entrada. Se um sombreador esperar mais entradas do que o número definido automaticamente, defina valores para essas entradas. Para obter detalhes sobre as diferentes maneiras de uso de um sombreador, e se os valores de entrada são definidos automaticamente, consulte "Uso de um sombreador" na página 318.

Especificação de valores de parâmetros de sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Alguns sombreadores definem valores de parâmetros que o sombreador usa para criar seu resultado. Por exemplo, um sombreador que altera o brilho de uma imagem pode especificar um parâmetro de brilho que determina o quanto a operação afeta o brilho. Um único parâmetro definido em um sombreador pode esperar um único valor ou vários valores, de acordo com a definição do parâmetro no sombreador. Cada parâmetro definido em um sombreador é representado no ActionScript por um objeto ShaderParameter. O objeto ShaderParameter é uma propriedade da ocorrência de ShaderData na propriedade de dados do objeto Shader, conforme descrito em "Identificação de entradas e parâmetros de sombreador" na página 313. Por exemplo, suponha que um sombreador defina um parâmetro denominado `brightness` e que o sombreador esteja representado por um objeto Shader denominado `myShader`. Nesse caso, você acessa o ShaderParameter que corresponde ao parâmetro `brightness` usando o seguinte identificador:

```
myShader.data.brightness
```

Para definir um ou mais valores para o parâmetro, crie uma matriz do ActionScript que contenha o valor ou valores e atribua essa matriz à propriedade `value` do objeto ShaderParameter. A propriedade `value` é definida como uma ocorrência de Array porque é possível que um único parâmetro do sombreador exija vários valores. Mesmo que o parâmetro do sombreador espere um único valor, você deve delimitar o valor em um objeto Array para atribuí-lo à propriedade `ShaderParameter.value`. A listagem a seguir demonstra como definir um único valor como a propriedade `value`:

```
myShader.data.brightness.value = [75];
```

Se o código-fonte de Pixel Bender do sombreador definir um valor padrão para o parâmetro, uma matriz que contém o valor ou valores padrão será criada e atribuída à propriedade `value` do objeto ShaderParameter quando o objeto Shader for criado. Depois que a matriz for atribuída à propriedade `value` (inclusive se ela for a matriz padrão), o valor do parâmetro poderá ser alterado com a alteração do valor do elemento de matriz. Você não precisa criar uma nova matriz e atribuí-la à propriedade `value`.

O exemplo a seguir demonstra a configuração de um valor do parâmetro do sombreador no ActionScript. Neste exemplo o sombreador define um parâmetro denominado `color`. O parâmetro `color` é declarado como uma variável `float4` no código-fonte do Pixel Bender, o que significa que ela é uma matriz de quatro números de ponto flutuante. No exemplo, o valor do parâmetro `color` é alterado continuamente e, cada vez que ele é alterado, o sombreador é usado para desenhar um retângulo colorido na tela. O resultado é uma alteração de cor animada.

Nota: *O código para este exemplo foi escrito por Ryan Taylor. Obrigado por compartilhar este exemplo, Ryan. Para ver o portfólio de Ryan e ler seus artigos, acesse www.boostworthy.com/.*

O código ActionScript está centralizado em torno de três métodos:

- `init()`: no método `init()`, o código carrega o arquivo de código de bytes do Pixel Bender que contém o sombreador. Quando o arquivo é carregado, o método `onLoadComplete()` é chamado.
- `onLoadComplete()`: no método `onLoadComplete()`, o código cria o objeto Shader chamado `shader`. Ele também cria uma ocorrência de Sprite chamada `texture`. No método `renderShader()`, o código desenha o resultado do sombreador em `texture` uma vez por quadro.
- `onEnterFrame()`: o método `onEnterFrame()` é chamado uma vez por quadro, o que cria o efeito de animação. Nesse método, o código define o valor do parâmetro do sombreador como a nova cor e, em seguida, chama o método `renderShader()` para desenhar o resultado do sombreador como um retângulo.

- `renderShader()`: no método `renderShader()`, o código chama o método `Graphics.beginShaderFill()` para especificar um preenchimento do sombreador. Em seguida, ele desenha um retângulo cujo preenchimento é definido pela saída do sombreador (a cor gerada). Para obter mais informações sobre como usar um sombreador desta maneira, consulte "Uso de um sombreador como um preenchimento de desenho" na página 319.

Veja abaixo o código ActionScript para este exemplo. Use esta classe com a classe de aplicativo principal em um projeto somente ActionScript no Flash Builder ou como a classe do documento para o arquivo FLA no 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();
        }
    }
}
```

Este é o código-fonte para o kernel de sombreador ColorFilter, usado para criar o arquivo de código de bytes do Pixel Bender “ColorFilter.pbj”:

```
<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;
    }
}
```

Se você estiver usando um sombreador cujos parâmetros não estão documentados, poderá calcular quantos elementos de qual tipo devem ser incluídos na matriz, marcando a propriedade `type` do objeto ShaderParameter. A propriedade `type` indica o tipo de dado do parâmetro, conforme definido no próprio sombreador. Para obter uma lista de números e tipos de elementos esperados por cada tipo de parâmetro, consulte a listagem da propriedade `ShaderParameter.value` na Referência do ActionScript 3.0.

Cada objeto ShaderParameter também tem uma propriedade `index` que indica onde o parâmetro se ajusta na ordem dos parâmetros do sombreador. Além dessas propriedades, um objeto ShaderParameter pode ter propriedades adicionais que contêm valores de metadados fornecidos pelo autor do sombreador. Por exemplo, o autor pode especificar valores de metadados, como valores mínimos, máximos e padrão para um parâmetro. Todos os valores de metadados especificados pelo autor são adicionados ao objeto ShaderParameter como propriedades dinâmicas. Para examinar essas propriedades, use um loop `for..in` para executar um loop nas propriedades dinâmicas do objeto ShaderParameter para identificar seus metadados. O exemplo a seguir mostra como usar um loop `for..in` para identificar metadados de um objeto ShaderParameter. Cada valor de metadados é adicionado a uma ocorrência de Vector denominada `metadata`. Observe que este exemplo pressupõe que uma ocorrência de Shader denominada `myShader` já foi criada, e que é conhecida como tendo um parâmetro denominado `brightness`:

```
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
```

Uso de um sombreador

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Depois que o sombreador Pixel Bender está disponível no ActionScript como um objeto Shader, ele pode ser usado de várias maneiras:

- Preenchimento de desenho do sombreador: O sombreador define a parte de preenchimento de uma forma desenhada usando a API de desenho
- Modo de mesclagem: O sombreador define a mesclagem entre dois objetos de exibição sobrepostos
- Filtro: O sombreador define um filtro que modifica a aparência do conteúdo visual
- Processamento autônomo do sombreador: O processamento do sombreador é executado sem especificar o uso pretendido da saída. Como opção, o sombreador pode ser executado no plano de fundo, com o resultado disponível quando o processamento é concluído. Essa técnica pode ser usada para gerar dados de bitmap e também para processar dados não visuais.

Uso de um sombreador como um preenchimento de desenho

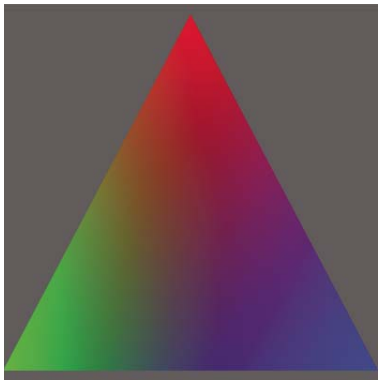
Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Ao usar um sombreador para criar um preenchimento de desenho, você deve utilizar os métodos de API de desenho para criar uma forma vetorial. A saída do sombreador é usada para preencher a forma, da mesma maneira que se pode usar qualquer imagem de bitmap como preenchimento de bitmap com a API de desenho. Para criar um preenchimento de sombreador, no ponto do código em que você deseja começar a desenhar a forma, chame o método `beginShaderFill()` do objeto Graphics. Passe o objeto Shader como o primeiro argumento do método `beginShaderFill()`, conforme mostrado nesta listagem:

```
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
```

Quando você usa um sombreador como preenchimento de desenho, deve definir quaisquer valores de imagem de entrada e valores de parâmetro exigidos pelo sombreador.

O exemplo a seguir demonstra o uso de um sombreador como preenchimento de desenho. Neste exemplo, o sombreador cria um gradiente de três pontas. Esse gradiente tem três cores, cada uma na ponta de um triângulo, com uma mesclagem de gradiente entre elas. Além disso, as cores se revezam para criar um efeito animado de cores girando.



Nota: *O código deste exemplo foi escrito por Petri Leskinen. Obrigado por compartilhar este exemplo, Petri. Para ver mais exemplos e tutoriais de Petri, acesse http://pixelero.wordpress.com/.*

O código ActionScript está baseado em três métodos:

- `init()`: o método `init()` é chamado quando o aplicativo é carregado. Neste método, o código define os valores iniciais dos objetos Point que representam as pontas do triângulo. O código também cria uma ocorrência de Sprite chamada `canvas`. Posteriormente, no método `updateShaderFill()`, o código desenha o resultado do sombreador em `canvas` uma vez por quadro. Por último, o código carrega o arquivo de código de bytes do sombreador.
- `onLoadComplete()`: no método `onLoadComplete()`, o código cria o objeto Shader chamado `shader`. Ele também define os valores de parâmetro iniciais. Para terminar, o código adiciona o método `updateShaderFill()` como um ouvinte do evento `enterFrame`, o que significa que ele é chamado uma vez por quadro para criar um efeito de animação.

- `updateShaderFill()`: o método `updateShaderFill()` é chamado uma vez por quadro, o que cria o efeito de animação. Neste método, o código calcula e define os valores dos parâmetros do sombreador. O código então chama o método `beginShaderFill()` para criar um preenchimento de sombreador e chama outros métodos de API de desenho para desenhar o resultado do sombreador em um triângulo.

Veja abaixo o código ActionScript para este exemplo. Use esta classe com a classe de aplicativo principal em um projeto somente ActionScript no Flash Builder ou como a classe do documento para um arquivo FLA no 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();
        }
    }
}
```

Este é o código-fonte do núcleo de sombreador ThreePointGradient, usado para criar o arquivo de código de bytes do Pixel Bender “ThreePointGradient.pbj”:

```
<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);
    }
}
```

Nota: *Se você usar um preenchimento por sombreamento ao renderizar com a unidade de processamento gráfico (GPU), a área preenchida assumirá a cor ciano.*

Para obter mais informações sobre como desenhar formas usando a API de desenho, consulte "Uso da API de desenho" na página 226.

Uso de um sombreador como modo de mesclagem

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O uso de um sombreador como modo de mesclagem é parecido com o uso de outros modos de mesclagem. O sombreador define a aparência resultante da mesclagem visual de dois objetos de exibição. Para usar um sombreador como modo de mesclagem, atribua o objeto Shader à propriedade `blendShader` do objeto de exibição em primeiro plano. A atribuição de um valor diferente de `null` à propriedade `blendShader` automaticamente define a propriedade `blendMode` do objeto de exibição como `BlendMode.SHADER`. A listagem a seguir demonstra o uso de um sombreador como modo de mesclagem. Observe que este exemplo pressupõe a existência de um objeto de exibição chamado `foreground` contido no mesmo pai na lista de exibição que outro conteúdo de exibição, com `foreground` se sobrepondo ao outro conteúdo:

```
foreground.blendShader = myShader;
```

Quando você usa um sombreador como modo de mesclagem, ele deve ser definido com pelo menos duas entradas. Como mostra o exemplo, você não define os valores de entrada no código. Em vez disso, as duas imagens mescladas são usadas automaticamente como entradas do sombreador. A imagem em primeiro plano é definida como a segunda imagem. (Este é o objeto de exibição ao qual o modo de mesclagem é aplicado.) Uma imagem em segundo plano é criada colocando-se a composição de todos os pixels atrás da caixa delimitadora da imagem em primeiro plano. Essa imagem em segundo plano é definida como a primeira imagem de entrada. Se usar um sombreador que espera mais de duas entradas, especifique um valor para qualquer entrada além das duas primeiras.

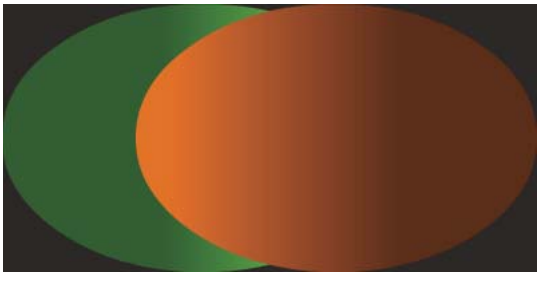
O exemplo a seguir demonstra o uso de um sombreador como modo de mesclagem. Este exemplo utiliza um modo de mesclagem de clareamento baseado na luminosidade. O resultado da mesclagem é que o valor de pixel mais claro de qualquer um dos objetos mesclados se torna o pixel que é exibido.

Nota: *O código deste exemplo foi escrito por Mario Klingemann. Obrigado por compartilhar este exemplo, Mario. Para conhecer mais sobre o trabalho de Mario e ler seus artigos, acesse www.quasimondo.com/.*

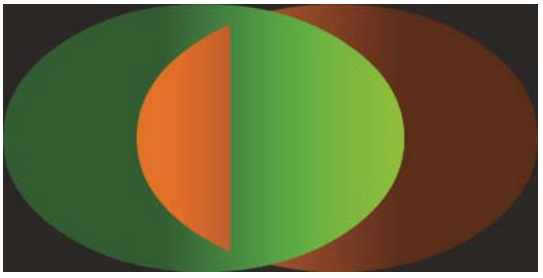
O código ActionScript importante está nestes dois métodos:

- `init()`: o método `init()` é chamado quando o aplicativo é carregado. Neste método, o código carrega o arquivo de código de bytes do sombreador.
- `onLoadComplete()`: no método `onLoadComplete()`, o código cria o objeto Shader chamado `shader`. Em seguida, ele desenha três objetos. O primeiro, `backdrop`, é um plano de fundo cinza escuro atrás de objetos mesclados. O segundo, `backgroundShape`, é uma elipse gradiente verde. O terceiro objeto, `foregroundShape`, é uma elipse gradiente cor de laranja.

  A elipse `foregroundShape` é o objeto de primeiro plano da mesclagem. A imagem de fundo da mesclagem é formada pela parte de `backdrop` e pela parte de `backgroundShape` que são sobrepostas pela caixa delimitadora do objeto `foregroundShape`. O objeto `foregroundShape` é o objeto mais à frente na lista de exibição. Ele se sobrepõe parcialmente a `backgroundShape` e completamente a `backdrop`. Por causa dessa sobreposição, sem um modo de mesclagem aplicado, a elipse laranja (`foregroundShape`) é exibida por completo e parte da elipse verde (`backgroundShape`) é ocultada por ela:



No entanto, com o modo de mesclagem aplicado, a parte mais brilhante da elipse verde "transparece" porque é mais clara do que a parte de `foregroundShape` que se sobrepõe a ela:



Veja abaixo o código ActionScript para este exemplo. Use esta classe com a classe de aplicativo principal em um projeto somente ActionScript no Flash Builder ou como a classe do documento para o arquivo FLA no 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;
        }
    }
}
```

Este é o código-fonte para o kernel de sombreador LumaLighten, usado para criar o arquivo de código de bytes do Pixel Bender “LumaLighten.pbj”:

```
<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;
    }
}
```

Para obter mais informações sobre como usar modos de mesclagem, consulte "Aplicação de modos de mesclagem" na página 189.

Nota: *Quando um programa Pixel Bender shader é executado como mescla no Flash Player ou no AIR, as funções de amostragem e* `outCoord()` *comportam-se de forma diferente de outros contextos. Em uma mescla, a função de amostragem sempre retornará o pixel atual avaliado pelo shader. Por exemplo, não é possível adicionar um deslocamento em* `outCoord()` *para criar uma amostra do pixel vizinho. Da mesma forma, se você utilizar a função*`outCoord()` *fora de uma função de amostragem, as coordenadas serão avaliadas em 0. Não é possível, por exemplo, utilizar a posição de um pixel para influenciar como as imagens mescladas serão combinadas.*

Uso de um sombreador como filtro

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Usar um sombreador como filtro é parecido com usar qualquer um dos outros filtros do ActionScript. Quando você utiliza um sombreador como filtro, a imagem filtrada (um objeto de exibição ou objeto BitmapData) é passada para ele. O sombreador usa a imagem de entrada para criar a saída do filtro, que geralmente é uma versão modificada da imagem original. Se o objeto filtrado é um objeto de exibição, a saída do sombreador é exibida na tela no lugar do objeto de exibição filtrado. Se o objeto filtrado é um objeto BitmapData, a saída do sombreador torna-se o conteúdo do objeto BitmapData cujo método `applyFilter()` é chamado.

Para usar um sombreador como filtro, primeiro você deve criar o objeto Shader conforme descrito em "Carregamento ou incorporação de um sombreador" na página 310. Em seguida, crie um objeto ShaderFilter vinculado ao objeto Shader. O objeto ShaderFilter é o filtro aplicado ao objeto filtrado. Você deve aplicá-lo a um objeto da mesma maneira que aplica qualquer filtro. Passe-o para a propriedade `filters` de um objeto de exibição ou chame o método `applyFilter()` em um objeto BitmapData. Por exemplo, o código a seguir cria um objeto ShaderFilter e aplica o filtro a um objeto de exibição chamado `homeButton`.

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
homeButton.filters = [myFilter];
```

Quando você usa um sombreador como filtro, o filtro deve ser definido com pelo menos uma entrada. Como mostra o exemplo, você não define o valor de entrada no código. Em vez disso, o objeto de exibição filtrado ou o objeto BitmapData é definido como a imagem de entrada. Se usar um sombreador que espera mais de uma entrada, especifique um valor para qualquer entrada além da primeira.

Em alguns casos, um filtro altera as dimensões da imagem original. Por exemplo, um típico efeito de sombra adiciona pixels extra contendo a sombra que é acrescentada à imagem. Quando usar um sombreador que altera as dimensões da imagem, defina as propriedades `leftExtension`, `rightExtension`, `topExtension` e `bottomExtension` para indicar em quanto você deseja que o tamanho da imagem seja alterado.

O exemplo a seguir demonstra o uso de um sombreador como filtro. O filtro deste exemplo inverte os valores de canal de vermelho, verde e azul de uma imagem. O resultado é a versão "negativa" da imagem.

Nota: *O sombreador usado neste exemplo é o kernel do Pixel Bender invertRGB.pbk fornecido com o Pixel Bender Toolkit. É possível carregar o código-fonte do kernel a partir do diretório de instalação do Pixel Bender Toolkit. Compile o código-fonte e salve o arquivo de código de bytes no mesmo diretório do código-fonte.*

O código ActionScript importante está nestes dois métodos:

- `init()`: o método `init()` é chamado quando o aplicativo é carregado. Neste método, o código carrega o arquivo de código de bytes do sombreador.
- `onLoadComplete()`: no método `onLoadComplete()`, o código cria o objeto Shader chamado `shader`. Em seguida, ele cria e desenha o conteúdo de um objeto chamado `target`. O objeto `target` é um retângulo preenchido com uma cor gradiente linear que é vermelha à esquerda, verde e amarela no meio e azul claro à direita. O objeto não filtrado é semelhante a este:

Com o filtro aplicado, as cores são invertidas, e o retângulo fica parecido com o seguinte:

O sombreador usado neste exemplo é o kernel do Pixel Bender de exemplo "invertRGB.pbk" fornecido com o Pixel Bender Toolkit. O código-fonte está disponível no arquivo "invertRGB.pbk" localizado no diretório de instalação do Pixel Bender Toolkit. Compile o código-fonte e salve o arquivo de código de bytes com o nome "invertRGB.pbj" no mesmo diretório que o seu código-fonte do ActionScript.

Veja abaixo o código ActionScript para este exemplo. Use esta classe com a classe de aplicativo principal em um projeto somente ActionScript no Flash Builder ou como a classe do documento para o arquivo FLA no 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];
        }
    }
}
```

Para obter mais informações sobre como aplicar filtros, consulte “Criação e aplicação de filtros” na página 275.

Uso de um sombreador no modo autônomo

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Quando você usa um sombreador no modo autônomo, o processamento do sombreador é executado independentemente do modo como você pretende usar a saída. Você especifica o sombreador a ser executado, define valores de entrada e de parâmetro e designa um objeto em que são colocados os dados resultantes. Você pode usar um sombreador no modo autônomo por dois motivos:

- Processamento de dados não de imagem: no modo autônomo, você pode optar por passar dados binários ou numéricos arbitrários para o sombreador em vez de dados de imagem de bitmap. Você pode determinar que o resultado do sombreador seja retornado como dados binários ou numéricos além dos dados de imagem de bitmap.
- Processamento em segundo plano: Quando você executa um sombreador no modo autônomo, por padrão ele é executado de maneira assíncrona. Isso significa que o sombreador é executado em segundo plano enquanto o aplicativo continua a executar, e o código recebe uma notificação quando o processamento do sombreador é concluído. Você pode usar um sombreador que demora bastante tempo para ser executado e não congela a interface de usuário do aplicativo ou outro processamento enquanto o sombreador está em execução.

Use um objeto ShaderJob para executar um sombreador no modo autônomo. Primeiro você deve criar o objeto ShaderJob e vinculá-lo ao objeto Shader que representa o sombreador a ser executado:

```
var job:ShaderJob = new ShaderJob(myShader);
```

Em seguida, defina quaisquer valores de entrada ou de parâmetro esperados pelo sombreador. Se estiver executando o sombreador em segundo plano, você também deverá registrar um ouvinte para o evento `complete` do objeto ShaderJob. O ouvinte será chamado quando o sombreador concluir seu trabalho:

```
function completeHandler(event:ShaderEvent):void
{
    // do something with the shader result
}

job.addEventListener(ShaderEvent.COMPLETE, completeHandler);
```

Em seguida, crie um objeto no qual o resultado da operação do sombreador será gravado quando a operação for concluída. Atribua esse objeto à propriedade `target` do objeto ShaderJob:

```
var jobResult:BitmapData = new BitmapData(100, 75);
job.target = jobResult;
```

Atribua uma ocorrência de BitmapData à propriedade `target`, caso você esteja usando ShaderJob para executar o processamento da imagem. Se estiver processando dados binários ou numéricos, atribua um objeto ByteArray ou uma ocorrência de Vector.<Number> para a propriedade `target`. Nesse caso, você deve definir as propriedades `width` e `height` do objeto ShaderJob para especificar o volume de dados a ser gerado para o objeto `target`.

Nota: *É possível definir as propriedades `shader`, `target`, `width` e `height` do objeto ShaderJob em uma única etapa passando argumentos para o construtor `ShaderJob()`, da seguinte forma:`var job:ShaderJob = new ShaderJob(myShader, myTarget, myWidth, myHeight);`*

Quando estiver pronto para executar o sombreador, chame o método `start()` do objeto ShaderJob:

```
job.start();
```

Por padrão, chamar `start()` faz com que ShaderJob seja executado de maneira assíncrona. Nesse caso, a execução do programa prossegue imediatamente com a próxima linha de código, em vez de esperar a conclusão do sombreador. Quando a operação do sombreador é concluída, o objeto ShaderJob chama seus ouvintes de evento `complete`, notificando-os sobre o término da execução. Nesse momento (isto é, no corpo do ouvinte de evento `complete`), o objeto `target` contém o resultado da operação do sombreador.

Nota: *Em vez de usar o objeto da propriedade `target`, você pode recuperar o resultado do sombreador diretamente do objeto do evento que é passado para o método do ouvinte. O objeto de evento é uma ocorrência de ShaderEvent. O objeto ShaderEvent tem três propriedades que podem ser usadas para acessar o resultado, dependendo do tipo de dados do objeto definido como a propriedade `target`: `ShaderEvent.bitmapData`, `ShaderEvent.byteArray` e `ShaderEvent.vector`.*

Se preferir, você pode passar um argumento `true` para o método `start()`. Nesse caso, a operação do sombreador será executada de maneira síncrona. Todo o código (inclusive a interação com a interface de usuário e quaisquer outros eventos) é pausado enquanto o sombreador é executado. Quando o sombreador é concluído, o objeto `target` contém o resultado do sombreador e o programa prossegue com a próxima linha de código.

```
job.start(true);
```

Capítulo 16: Trabalho com clipes de filme

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe MovieClip é a classe principal para símbolos de animação e clipe de vídeo que você cria no seu ambiente de desenvolvimento do Adobe® Flash®. Ela possui todos os comportamentos e funcionalidades dos objetos de exibição, mas com propriedades e métodos adicionais para controlar sua linha de tempo.

Noções básicas de clipes de filme

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os clipes de filme são elementos essenciais para quem cria conteúdo animado com a ferramenta de autoria do Flash e deseja controlar esse conteúdo com o ActionScript. Sempre que você cria um símbolo de clipe de filme no Flash, o Flash adiciona o símbolo à biblioteca desse documento Flash. Por padrão, esse símbolo se torna uma ocorrência da Classe MovieClip e, como tal, tem as propriedades e métodos da classe MovieClip.

Quando uma ocorrência de um símbolo de clipe de filme é colocada no Palco, o clipe de filme avança automaticamente na linha de tempo (se tiver mais de um quadro), a menos que a reprodução seja alterada usando o ActionScript. É essa linha de tempo que distingue a classe MovieClip, permitindo que você crie a animação por meio de interpolações de movimento ou forma com a ferramenta de autoria do Flash. Entretanto, com um objeto de exibição que é uma ocorrência da classe Sprite, você pode criar a animação apenas alterando os valores do objeto de modo programático.

Nas versões anteriores do ActionScript, a classe MovieClip era a base para todas as ocorrências no Palco. No ActionScript 3.0, um clipe de filme é apenas um dos muitos objetos que podem aparecer na tela. Se uma linha de tempo não for necessária para a função de um objeto de exibição, o uso da classe Shape ou Sprite no lugar da classe MovieClip poderá melhorar o desempenho de renderização. Para obter informações sobre a escolha do objeto de exibição apropriado para uma tarefa, consulte “Escolha de uma subclasse de DisplayObject” na página 174.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes relacionados aos clipes de filme:

SWF AVM1 Um arquivo SWF criado usando o ActionScript 1.0 ou o ActionScript 2.0, normalmente destinado à executado no Flash Player 8 ou anterior.

SWF AVM2 Um arquivo SWF criado usando o ActionScript 3.0 para Adobe Flash Player 9 ou posterior ou Adobe AIR.

SWF externo Um arquivo SWF que é criado separadamente do arquivo SWF de projeto e serve para ser carregado no arquivo SWF de projeto e reproduzido nesse arquivo SWF.

Quadro A menor divisão de tempo na linha de tempo. Assim como em um fotograma de filme de cinema, cada quadro é como um instantâneo da animação no tempo e, quando os quadros são reproduzidos rapidamente em sequência, o efeito da animação é criado.

Linha do tempo A representação metafórica da série de quadros forma a sequência de animação de um clipe de filme. A linha de tempo de um objeto MovieClip é equivalente à da ferramenta de autoria do Flash.

Indicador de reprodução Um marcador identificando o local (quadro) na linha de tempo que está sendo exibido em um determinado momento.

Trabalho com objetos MovieClip

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao publicar um arquivo SWF, o Flash converte todas as ocorrências do símbolo de clipe de filme no Palco para objetos MovieClip. Para disponibilizar um símbolo de clipe de filme para o ActionScript, dê a ele um nome de ocorrência no campo Nome da ocorrência do Inspetor de propriedades. Quando um arquivo SWF é criado, o Flash gera o código que cria a ocorrência de MovieClip no Palco e declara uma variável usando o nome da ocorrência. Se você nomeou clipes de filme que estão aninhados em outros clipes de filme nomeados, esses clipes de filme filho serão tratados como propriedades do clipe de filme pai (você pode acessar o clipe de filme filho usando a sintaxe de pontos). Por exemplo, se um clipe de filme com o nome de ocorrência `childClip` estiver aninhado em outro clipe com o nome de ocorrência `parentClip`, você poderá reproduzir a animação de linha de tempo do clipe filho chamando este código:

```
parentClip.childClip.play();
```

Nota: *: Ocorrências-filho inseridas no Palco, na ferramenta de autoria do Flash, não podem ser acessadas pelo código a partir do construtor de uma ocorrência-pai, já que não foram criadas nesse ponto da execução do código. Antes de acessar o filho, o pai deve, em vez disso, criar a ocorrência-filho por código ou atrasar o acesso a uma função de retorno de chamada que ouve o filho de modo a despachar seu evento* `Event.ADDED_TO_STAGE`*.*

Embora alguns métodos e propriedades herdados da classe MovieClip do ActionScript 2.0 permaneçam os mesmos, outros foram alterados. Todas as propriedades prefixadas com um sublinhado foram renomeadas. Por exemplo, as propriedades `_width` e `_height` agora são acessadas como `width` e `height`, enquanto que `_xscale` e `_yscale` agora são acessadas como `scaleX` e `scaleY`. Consulte uma lista completa das propriedades e dos métodos da classe MovieClip em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Controle de reprodução de clipe de filme

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash usa a metáfora de uma linha de tempo para transmitir uma animação ou uma alteração de estado. Qualquer elemento visual que empregue uma linha de tempo deve ser um objeto MovieClip ou uma extensão da classe MovieClip. Embora o ActionScript possa instruir qualquer clipe de filme a parar, reproduzir ou ir para outro ponto na linha de tempo, ele não pode ser usado para criar dinamicamente uma linha de tempo ou adicionar conteúdo a quadros específicos; isso só é possível usando a ferramenta de autoria do Flash.

Ao ser reproduzido, o MovieClip avança na linha de tempo em uma velocidade controlada pela velocidade de projeção do arquivo SWF. Se desejar, você pode substituir essa configuração definindo a propriedade `Stage.frameRate` no ActionScript.

Reproduzir clipes de filme e parar a reprodução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os métodos `play()` e `stop()` permitem o controle básico de um clipe de filme na linha de tempo. Por exemplo, suponha que você tenha um símbolo de clipe de filme no Palco que contém uma animação de uma bicicleta se movendo pela tela, com o nome de ocorrência definido como `bicycle`. Se o seguinte código for anexado a um quadro-chave na linha de tempo principal,

```
bicycle.stop();
```

a bicicleta não se moverá (a animação não será reproduzida). O movimento da bicicleta pode começar por meio de outra interação do usuário. Por exemplo, se você tivesse um botão chamado `startButton`, o seguinte código em um quadro-chave na linha de tempo principal faria a animação ser reproduzida com um clique no botão:

```
// 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);
```

Avançar e retroceder

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os métodos `play()` e `stop()` não são a única forma de controlar a reprodução em um clipe de filme. Você também pode mover o indicador de reprodução para frente ou para trás na linha de tempo manualmente, usando os métodos `nextFrame()` e `prevFrame()`. Quando qualquer um desses métodos é chamado, a reprodução para e move o indicador de reprodução um quadro para frente ou para trás, respectivamente.

Usar o método `play()` é como chamar `nextFrame()` sempre que o evento `enterFrame` do objeto do clipe de filme é disparado. Com essas linhas, você pode fazer a reprodução do clipe de filme `bicycle` retroceder, criando um ouvinte de eventos para o evento `enterFrame` e instruindo `bicycle` a ir para o quadro anterior na função do ouvinte, desta forma:

```
// 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);
```

Na reprodução normal, se um clipe de filme tiver mais de um quadro, ele fará loop indefinidamente ao ser reproduzido, ou seja, ele retornará ao Quadro 1 quando ultrapassar o quadro final. Ao usar `prevFrame()` ou `nextFrame()`, esse comportamento não acontece automaticamente (chamar `prevFrame()` quando o indicador de reprodução está no Quadro 1 não o move para o último quadro). A condição `if` no exemplo anterior verifica se o indicador de reprodução retrocedeu para o primeiro quadro e o define à frente de seu quadro final, criando assim um loop contínuo do clipe de filme cuja reprodução retrocede.

Salto para um quadro diferente e uso de rótulos de quadro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O envio de um clipe de filme para um novo quadro é simples. A chamada de `gotoAndPlay()` ou `gotoAndStop()` fará o clipe de filme saltar para o número de quadro especificado como parâmetro. Se preferir, você pode transmitir uma sequência de caracteres que corresponda ao nome de um rótulo de quadro. É possível atribuir um rótulo a qualquer quadro na linha de tempo. Para fazer isso, selecione um quadro na linha de tempo e digite um nome no campo Rótulo do quadro no Inspetor de propriedades.

As vantagens de usar rótulos de quadro em vez de números ficam mais evidentes ao criar um clipe de filme complexo. Quando o número de quadros, camadas e interpolações em uma animação for grande, considere a rotulagem de quadros importantes com descrições explicativas que representem mudanças de comportamento no clipe de filme (por exemplo, "off", "walking", "running"). Isso melhora a confiabilidade do código e também fornece flexibilidade, já que as chamadas do ActionScript que vão para um quadro rotulado são ponteiros para uma única referência, o rótulo, em vez de um número de quadro específico. Se mais tarde você decidir mover um segmento específico da animação para um quadro diferente, não será necessário alterar o código do ActionScript contanto que mantenha o mesmo rótulo para os quadros no novo local.

Para representar os rótulos de quadro no código, o ActionScript 3.0 inclui a classe FrameLabel. Cada ocorrência dessa classe representa um único rótulo de quadro e tem uma propriedade `name` representando o nome do rótulo de quadro conforme especificado no Inspetor de propriedades e uma propriedade`frame` representando o número do quadro com rótulo colocado na linha de tempo.

Para obter acesso às ocorrências de FrameLabel associadas a uma ocorrência de clipe de filme, a classe MovieClip inclui duas propriedades que retornam objetos FrameLabel diretamente. A propriedade `currentLabels` retorna uma matriz que consiste em todos os objetos FrameLabel na linha de tempo inteira de um clipe de filme. A propriedade `currentLabel` retorna uma sequência de caracteres contendo o nome do rótulo de quadro encontrado mais recentemente na linha de tempo.

Suponha que você crie um clipe de filme chamado `robot` e rotulou os diversos estados dessa animação. Você pode configurar uma condição que verifica a propriedade `currentLabel` para acessar o estado atual de `robot`, como no seguinte código:

```
if (robot.currentLabel == "walking")
{
    // do something
}
```

O Flash Player 11.3 e o AIR 3.3 adicionaram o evento `frameLabel` à classe FrameLabel. Você pode atribuir um manipulador de eventos à instância FrameLabel que representa um rótulo de quadro. O evento é despachado quando o indicador de reprodução entra no quadro.

O exemplo a seguir cria uma instância FrameLabel para o segundo rótulo de quadro na Matriz de rótulos de quadro do MovieClip. Em seguida, registra um manipulador de eventos para o evento `frameLabel`:

```
var myFrameLabel:FrameLabel = robot.currentLabels[1];
myFrameLabel.addEventListener(Event.FRAME_LABEL, onFrameLabel);

function onFrameLabel(e:Event):void {
    //do something
}
```

Trabalho com cenas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ambiente de autoria do Flash, você pode usar cenas para demarcar uma série de linhas de tempo pelas quais um arquivo SWF pode avançar. Com o uso do segundo parâmetro dos métodos `gotoAndPlay()` ou `gotoAndStop()`, é possível especificar uma cena à qual enviar o indicador de reprodução. Todos os arquivos FLA começam apenas com a cena inicial, mas você pode criar novas cenas.

O uso de cenas nem sempre é a melhor abordagem porque as cenas apresentam várias desvantagens. Um documento Flash contendo várias cenas pode ser difícil de manter, principalmente em ambientes de vários autores. Várias cenas também podem ser ineficientes em largura de banda, porque o processo de publicação mescla todas as cenas em uma única linha de tempo. Isso provoca um download progressivo de todas as cenas, mesmo que elas nunca sejam reproduzidas. Por esses motivos, o uso de várias cenas, muitas vezes, não é recomendado para organizar várias animações longas baseadas na linha de tempo.

A propriedade `scenes` da classe MovieClip é uma matriz de objetos Scene que representa todas as cenas no arquivo SWF. A propriedade `currentScene` retorna um objeto Scene que representa a cena que está em execução no momento.

A classe Scene possui várias propriedades que fornecem informações sobre uma cena. A propriedade `labels` retorna uma matriz de objetos FrameLabel representando os rótulos de quadro dessa cena. A propriedade `name` retorna o nome da cena como uma sequência de caracteres. A propriedade `numFrames` retorna um int representando o número total de quadro na cena.

Criação de objetos MovieClip com o ActionScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma maneira de adicionar conteúdo à tela no Flash é arrastando ativos da biblioteca para o Palco, mas esse não é o único fluxo de trabalho. Para projetos complexos, os desenvolvedores experientes em geral preferem criar clipes de filme de modo programático. Essa abordagem apresenta várias vantagens: facilidade de reutilização de código, velocidade de tempo de compilação mais rápida e modificações mais sofisticadas que estão disponíveis apenas para o ActionScript.

A API da lista de exibição do ActionScript 3.0 simplifica o processo de criar dinamicamente os objetos MovieClip. A capacidade de instanciar uma ocorrência de MovieClip diretamente, separada do processo de adicioná-la á lista de exibição, fornece flexibilidade e simplicidade sem sacrificar o controle.

No ActionScript 3.0, quando você cria uma ocorrência de clipe de filme (ou qualquer outro objeto de exibição) de modo programático, ela só aparece na tela quando é adicionada à lista de exibição, chamando `addChild()` ou o método `addChildAt()` em um contêiner de objetos de exibição. Isso permite criar um clipe de filme, definir suas propriedades e até chamar métodos antes que ele seja renderizado na tela. Para obter informações sobre como trabalhar com a lista de exibição, consulte "Trabalho com contêineres de objeto de exibição" na página 161.

Exportação de símbolos da biblioteca para o ActionScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Por padrão, as ocorrências de símbolos de clipe de filme em uma biblioteca de documentos Flash não podem ser criadas dinamicamente (ou seja, usando apenas o ActionScript). Isso porque cada símbolo que é exportado para ser usado no ActionScript aumenta o tamanho do arquivo SWF, e talvez não haja a intenção de usar alguns símbolos no palco. Por isso, para que um símbolo se torne disponível, é necessário especificar que ele seja exportado para o ActionScript.

Para exportar um símbolo para o ActionScript:

1. Selecione o símbolo no painel Biblioteca e abra a caixa de diálogo Propriedades do símbolo.
2. Se necessário, ative as Configurações avançadas.
3. Na seção de ligação, marque a caixa de seleção Exportar para ActionScript.

   Isso ativará os campos Classe e Classe base.

   Por padrão, o campo Classe é preenchido com o nome do símbolo, com os espaços removidos (por exemplo, um símbolo chamado "Tree House" se tornaria "TreeHouse"). Para especificar que o símbolo use uma classe personalizada para seu comportamento, digite o nome completo da classe incluindo seu pacote neste campo. Se pretende criar ocorrências do símbolo no ActionScript, mas não precisa adicionar nenhum comportamento adicional, você pode deixar o nome da classe como está.

   O valor da Classe base assume `flash.display.MovieClip` como padrão. Se quiser estender a funcionalidade do símbolo para outra classe personalizada, você poderá especificar o nome dessa classe, contanto que ela estenda a classe Sprite (ou MovieClip).
4. Pressione o botão OK para salvar as alterações.

   Nesta etapa, se o Flash não encontrar um arquivo SWC vinculado ou um arquivo ActionScript externo com uma definição para a classe especificada (por exemplo, caso não seja necessário adicionar comportamentos extras para o símbolo), um aviso é exibido:

   Não foi possível localizar uma definição para esta classe no caminho de classe. Portanto, uma definição será automaticamente gerada no arquivo SWF após a exportação..

   Desconsidere esse aviso se o símbolo da biblioteca não exigir funcionalidade exclusiva além daquela da classe MovieClip.

Se você não fornecer uma classe para o símbolo, o Flash criará uma para ele equivalente a esta:

```
package
{
    import flash.display.MovieClip;

    public class ExampleMovieClip extends MovieClip
    {
        public function ExampleMovieClip()
        {
        }
    }
}
```

Se quiser adicionar outra funcionalidade ActionScript ao símbolo, adicione as propriedades e métodos adequados à estrutura de código abaixo. Por exemplo, suponha que você tenha um símbolo de clipe de filme contendo um círculo com 50 pixels de largura e 50 de altura, e o símbolo seja especificado para ser exportado para o ActionScript com uma classe chamada Circle. O código a seguir, quando colocado em um arquivo Circle.as, estende a classe MovieClip e fornece ao símbolo os métodos adicionais `getArea()` e `getCircumference()`:

```
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;
        }
    }
}
```

O seguinte código, colocado em um quadro-chave no Quadro 1 do documento Flash, criará uma ocorrência do símbolo e a exibirá na tela:

```
var c:Circle = new Circle();
addChild(c);
trace(c.width);
trace(c.height);
trace(c.getArea());
trace(c.getCircumference());
```

Esse código demonstra a instanciação baseada no ActionScript como uma alternativa para arrastar ativos individuais ao Palco. Isso cria um círculo com todas as propriedades de um clipe de filme e com os métodos personalizados definidos na classe Circle. Este exemplo é bem básico; o símbolo da biblioteca pode especificar um número qualquer de propriedades e métodos em sua classe.

A instanciação baseada no ActionScript é muito eficiente, porque permite criar dinamicamente grandes quantidades de ocorrências, um trabalho que seria cansativo se feito manualmente. Ela também é flexível, porque você pode personalizar as propriedades de cada ocorrência na sua criação. Para ter uma ideia dessas duas vantagens, use um loop para criar dinamicamente várias ocorrências de Circle. Com o símbolo e a classe Circle descritos anteriormente na sua biblioteca de documentos Flash, coloque o seguinte código em um quadro-chave do Quadro 1:

```
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);
}
```

Isso demonstra como você pode criar e personalizar várias ocorrências de um símbolo com rapidez usando código. Cada ocorrência é posicionada com base na contagem atual dentro do loop e recebe uma cor aleatória definindo sua propriedade `transform` (que Circle herda ao estender a classe MovieClip).

Carregamento de um arquivo SWF externo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 3.0, os arquivos SWF são carregados usando a classe Loader. Para carregar um arquivo SWF externo, o ActionScript precisa fazer quatro coisas:

1. Criar um novo objeto URLRequest com a url do arquivo.
2. Criar um novo objeto Loader.
3. Chamar o método `load()` do objeto Loader, transmitindo a ocorrência de URLRequest como um parâmetro.
4. Chame o método `addChild()` em um contêiner de objetos de exibição (como a linha de tempo principal de um documento Flash) para adicionar a ocorrência de Loader à lista de exibição.

Por fim, o código fica semelhante a este:

```
var request:URLRequest = new URLRequest("http://www.[yourdomain].com/externalSwf.swf");
var loader:Loader = new Loader()
loader.load(request);
addChild(loader);
```

Esse mesmo código pode ser usado para carregar um arquivo de imagem externo, como uma imagem JPEG, GIF ou PNG, especificando a url do arquivo de imagem em vez da url de um arquivo SWF. Um arquivo SWF, diferentemente de um arquivo de imagem, pode conter o ActionScript. Por isso, embora o processo de carregar um arquivo SWF seja idêntico ao de carregar uma imagem, no carregamento de um arquivo SWF externo, o arquivo SWF que faz o carregamento e o arquivo SWF que é carregado residem na mesma caixa de proteção caso o Flash Player ou o AIR executem o SWF e você pretenda usar o ActionScript para se comunicar de alguma forma com arquivo SWF. Além disso, se o arquivo SWF contiver classes que compartilhem o mesmo namespace das classes no arquivo SWF de carregamento, talvez seja necessário criar um novo domínio de aplicativo para o arquivo SWF carregado a fim de evitar conflitos de namespace. Para obter mais informações sobre segurança e considerações sobre domínio de aplicativo, consulte "Trabalhar com domínios de aplicativo" na página 149 e "Carregamento de conteúdo" na página 1098.

Quando carregado com êxito, o arquivo SWF externo pode ser acessado por meio da propriedade `Loader.content`. Se for publicado no ActionScript 3.0, o arquivo SWF externo será um clipe de filme ou uma entidade gráfica, dependendo da classe que ele estender.

Há algumas diferenças para carregar um arquivo SWF no Adobe AIR for iOS em contraste com outras plataformas. Para obter mais informações, consulte "Carregamento de arquivos SWF no AIR for iOS" na página 205.

Considerações sobre o carregamento de um arquivo SWF antigo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se o arquivo SWF externo for publicado com uma versão mais antiga do ActionScript, há importantes limitações a serem consideradas. Diferentemente de um arquivo SWF do ActionScript 3.0 que é executado com AVM2 (ActionScript Virtual Machine 2), um arquivo SWF publicado para o ActionScript 1.0 ou no 2.0 é executado com AVM1 (ActionScript Virtual Machine 1).

Há diferenças importantes ao carregar um arquivo SWF do ActionScript 1.0 ou 2.0 em um arquivo SWF do ActionScript 3.0 (em comparação com o carregamento de um arquivo SWF do ActionScript 3.0). O Flash Player fornece compatibilidade total com conteúdo publicado em versões anteriores. Qualquer conteúdo que é executado em versões anteriores do Flash Player pode ser reproduzido nas versões do Flash Player que suportam o ActionScript 3.0. No entanto, são aplicadas as seguintes limitações:

- O código do ActionScript 3.0 pode carregar um arquivo SWF escrito no ActionScript 1.0 ou 2.0. Quando um arquivo SWF do ActionScript 1.0 or 2.0 é carregado com sucesso, o objeto carregado (a propriedade `Loader.content`) é um objeto AVM1Movie. Uma ocorrência de AVM1Movie não é igual a uma ocorrência de MovieClip. É um objeto de exibição, mas, diferentemente de um clipe de filme, não inclui métodos ou propriedades relacionadas à linha de tempo. O arquivo SWF AVM2 pai não pode acessar as propriedades, métodos ou objetos do objeto AVM1Movie carregado.
- Os arquivos SWF escritos no ActionScript 1.0 ou 2.0 não podem carregar arquivos SWF escritos no ActionScript 3.0. Isso significa que os arquivos SWF criados no Flash 8 ou no Flex Builder 1.5 ou versões anteriores não podem carregar arquivos SWF do ActionScript 3.0.

  A única exceção a essa regra é que um arquivo SWF do ActionScript 2.0 pode ser substituído por um arquivo SWF do ActionScript 3.0, contanto que o arquivo SWF do Action Script 2.0 não tenha carregado nada antes em qualquer um de seus níveis. Um arquivo SWF do ActionScript 2.0 pode fazer isso por meio de uma chamada a `loadMovieNum()`, transmitindo um valor de 0 ao parâmetro `level`.
- Em geral, os arquivos SWF escritos no ActionScript 1.0 ou 2.0 devem ser migrados para trabalhar com os arquivos SWF escritos no ActionScript 3.0. Por exemplo, suponhamos que você criou um player de mídia usando o ActionScript 2.0. Ele carrega vários conteúdos que também foram criados usando o ActionScript 2.0. Se você criar um novo conteúdo no ActionScript 3.0, não poderá carregá-lo no player de mídia. Será necessário migrar o player de vídeo para o ActionScript 3.0.

Se, no entanto, você criar um player de mídia no ActionScript 3.0, ele poderá executar carregamentos simples do seu conteúdo do ActionScript 2.0.

As tabelas a seguir resumem as limitações das versões anteriores do Flash Player em relação ao carregamento de conteúdo mais novo e execução de código, bem como as limitações de cross-scripting entre arquivos SWF escritos em versões diferentes do ActionScript.

Funcionalidade com suporte	Flash Player 7	Flash Player 8	Flash Player 9 e 10
Pode carregar SWFs publicados para	7 e anterior	8 e anterior	9 (ou 10) e anterior
Contém este AVM	AVM1	AVM1	AVM1 e AVM2
Executa SWFs gravados no ActionScript	1.0 e 2.0	1.0 e 2.0	1.0, 2.0 e 3.0

Na tabela a seguir, "Funcionalidade com suporte" refere-se ao conteúdo executado no Flash Player 9 ou posterior. O conteúdo executado no Flash Player 8 ou anterior só pode ser carregar, exibir, executar e cruzar scripts no ActionScript 1.0 e 2.0.

Funcionalidade com suporte	Conteúdo criado no ActionScript 1.0 e 2.0	Conteúdo criado no ActionScript 3.0
Pode carregar conteúdo e executar código no conteúdo criado no	ActionScript 1.0 e 2.0 apenas	ActionScript 1.0, 2.0 e 3.0
Pode cruzar conteúdo de script criado no	ActionScript 1.0 e 2.0 apenas (ActionScript 3.0 até LocalConnection)	ActionScript 1.0 e 2.0 até LocalConnection. ActionScript 3.0

Exemplo de clipe de filme: RuntimeAssetsExplorer

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A funcionalidade Exportar para ActionScript pode ser especialmente vantajosa para bibliotecas que podem ser úteis em mais de um projeto. Se o Flash Player ou o AIR executar um arquivo SWF, os símbolos exportados para o ActionScript ficarão disponíveis para qualquer arquivo SWF dentro da mesma caixa de proteção de segurança que o SWF que o carregar. Dessa forma, um único documento Flash pode gerar um arquivo SWF exclusivamente designado para manter ativos gráficos. Essa técnica é especialmente útil para projetos grandes nos quais os designers que lidam com ativos visuais podem trabalhar em paralelo com os desenvolvedores que criam um arquivo SWF "empacotador" que carrega o arquivo SWF de ativos gráficos em tempo de execução. Você pode usar esse método para manter uma série de arquivos de versão nos quais os ativos gráficos não dependem do progresso do desenvolvimento de programação.

O aplicativo RuntimeAssetsExplorer carrega qualquer arquivo SWF que for uma subclasse de RuntimeAsset e permite navegar pelos ativos disponíveis desse arquivo SWF. O exemplo ilustra o seguinte:

- Carregamento de um arquivo SWF externo usando `Loader.load()`
- Criação dinâmica de um símbolo de biblioteca exportado para o ActionScript
- Controle do ActionScript de reprodução de MovieClip

Antes de começar, observe que cada arquivo SWF a ser executado no Flash Player deve estar localizado na mesma caixa de proteção de segurança. Para obter mais informações, consulte “Caixas de proteção de segurança” na página 1081.

Para obter os arquivos de aplicativos dessa amostra, efetue o download de Amostras do Flash Professional. Os arquivos de aplicativo RuntimeAssetsExplorer podem ser encontrados na pasta Samples/RuntimeAssetsExplorer. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
RuntimeAssetsExample.mxml ou RuntimeAssetsExample.fla	A interface do usuário do aplicativo para Flex (MXML) ou Flash (FLA).
RuntimeAssetsExample.as	Classe de documento para o aplicativo Flash (FLA).
GeometricAssets.as	Uma classe de exemplo que implementa a interface RuntimeAsset.
GeometricAssets.fla	Um arquivo FLA vinculado à classe GeometricAssets (a classe de documento do FLA) contendo símbolos que são exportados para o ActionScript.
com/example/programmingas3/runtimeassetexplorer/RuntimeLibrary.as	Uma interface que define os métodos necessários esperados de todos os arquivos SWF de ativos de tempo de execução que serão carregados no contêiner do explorador.
com/example/programmingas3/runtimeassetexplorer/AnimatingBox.as	A classe do símbolo de biblioteca na forma de uma caixa de rotação.
com/example/programmingas3/runtimeassetexplorer/AnimatingStar.as	A classe do símbolo de biblioteca na forma de uma estrela de rotação.

Estabelecimento de uma interface de biblioteca de tempo de execução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para que o explorador interaja adequadamente com uma biblioteca SWF, a estrutura das bibliotecas de ativos de tempo de execução deve ser formalizada. Faremos isso criando uma interface, que é semelhante a uma classe pois trata-se de um projeto de métodos que demarca uma estrutura esperada, mas, diferentemente de uma classe, não inclui nenhum corpo de método. A interface fornece um meio de comunicação ente a biblioteca de tempo de execução e o explorador. Cada SWF de ativos de tempo de execução que for carregado em nosso navegador implementará essa interface. Para obter informações sobre interfaces e como elas podem ser úteis, consulte Interfaces em *Aprendizado do ActionScript 3.0*.

A interface RuntimeLibrary será bem simples: precisamos apenas de uma função que forneça ao explorador uma matriz de caminhos de classe para os símbolos a serem exportados e disponibilizados na biblioteca de tempo de execução. Para esse fim, a interface possui um único método: `getAssets()`.

```
package com.example.programmingas3.runtimeassetexplorer
{
    public interface RuntimeLibrary
    {
        function getAssets():Array;
    }
}
```

Criação do arquivo SWF da biblioteca de ativos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao definir a interface RuntimeLibrary, é possível criar vários arquivos SWF de biblioteca de ativos que podem ser carregados em outro arquivo SWF. A criação de uma biblioteca SWF individual de ativos envolve quatro tarefas:

- Criação de uma classe para o arquivo SWF da biblioteca de ativos
- Criação de classes para ativos individuais contidos na biblioteca
- Criação dos ativos gráficos em si
- Associação de elementos gráficos a classes e publicação do SWF da biblioteca

Criação de uma classe para implementar a interface RuntimeLibrary

Em seguida, criaremos a classe GeometricAssets que implementará a interface RuntimeLibrary. Ela será a classe de documento do FLA. O código para essa classe é muito semelhante à interface RuntimeLibrary; a diferença entre eles é que, na definição da classe, o método `getAssets()` possui um corpo de método.

```
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" ];
        }
    }
}
```

Se fôssemos criar uma segunda biblioteca de tempo de execução, criaríamos outro FLA com base em outra classe (por exemplo, AnimationAssets) que fornece sua própria implementação de `getAssets()`.

Criação de classes para cada ativo MovieClip

Para este exemplo, iremos apenas estender a classe MovieClip sem adicionar nenhuma funcionalidade aos ativos personalizados. O seguinte código para AnimatingStar é análogo ao de AnimatingBox:

```
package com.example.programmingas3.runtimeassetexplorer
{
    import flash.display.MovieClip;

    public class AnimatingStar extends MovieClip
    {
        public function AnimatingStar() {
        }
    }
}
```

Publicação da biblioteca

Agora, iremos conectar os ativos baseados no MovieClip à nova classe, criando um novo FLA e inserindo GeometricAssets no campo Classe do documento do Inspetor de propriedades. Para fins deste exemplo, criaremos duas formas básicas que suam uma interpolação de linha de tempo para fazer uma rotação no sentido horário em 360 quadros. Os símbolos `animatingBox` e `animatingStar` são definidos para Exportar para ActionScript e o campo Classe é definido com os respectivos caminhos de classe especificados na implementação de `getAssets()`. A classe base padrão de `flash.display.MovieClip` permanece, pois desejamos subclassificar os métodos MovieClip padrão.

Depois de definir as configurações de exportação do símbolo, publique o FLA. Agora, sua primeira biblioteca de tempo de execução está pronta. Se esse arquivo SWF fosse carregado em outro arquivo SWF AVM2, os símbolos AnimatingBox e AnimatingStar ficariam disponíveis para o arquivo SWF.

Carregamento da biblioteca em outro arquivo SWF

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A última questão funcional a ser resolvida é a interface do usuário para o explorador de ativos. Neste exemplo, o caminho para a biblioteca de tempo de execução é codificado como uma variável chamada `ASSETS_PATH`. Se preferir, você pode usar a classe FileReference, por exemplo, para criar uma interface que navegue para um arquivo SWF específico do seu disco rígido.

Quando a biblioteca de tempo de execução é carregada com êxito, o Flash Player chama o método `runtimeAssetsLoadComplete()`:

```
private function runtimeAssetsLoadComplete(event:Event):void
{
    var rl:* = event.target.content;
    var assetList:Array = rl.getAssets();
    populateDropdown(assetList);
    stage.frameRate = 60;
}
```

Nesse método, a variável rl representa o arquivo SWF carregado. O código chama o método `getAssets()` do arquivo SWF carregado, obtendo uma lista de ativos que estão disponíveis e os usa para preencher um componente ComboBox com uma lista de ativos disponíveis, chamando o método `populateDropDown()`. Esse método por sua vez armazena o caminho de classe completo de cada ativo. Ao ser clicado, o botão Adicionar da interface do usuário dispara o método `addAsset()`:

```
private function addAsset():void
{
    var className:String = assetNameCbo.selectedItem.data;
    var AssetClass:Class = getDefinitionByName(className) as Class;
    var mc:MovieClip = new AssetClass();
    ...
}
```

que obtém o caminho de classe do ativo atualmente selecionado no ComboBox (`assetNameCbo.selectedItem.data`) e usa a função `getDefinitionByName()` (do pacote flash.utils) para obter uma referência real para a classe do ativo a fim de criar uma nova ocorrência desse ativo.

Capítulo 17: Trabalho com interpolações de movimento

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

"Animação de objetos" na página 196 descreve como implementar animações com script no ActionScript.

Neste capítulo descreveremos uma técnica diferente de criação de animações: a interpolação de movimento. Essa técnica permite criar um movimento ao definir o movimento interativamente em um documento que usa o Adobe® Flash® Professional. Você poderá usar o movimento recém-criado em sua animação dinâmica com base em ActionScript no tempo de execução.

O Flash Professional gera automaticamente o código ActionScript que implementa a interpolação de movimento e a disponibiliza para cópia e reutilização.

Para criar interpolações de movimento, é preciso ter a licença do Flash Professional.

Mais tópicos da Ajuda

pacote fl.motion

Noções básicas de interpolações de movimento

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

As interpolações de movimento facilitam a criação de animações.

Uma interpolação de movimento modifica as propriedades do objeto de exibição, tais como posição ou rotação, eu uma base quadro a quadro. Uma interpolação de movimento também pode mudar a aparência de um objeto de exibição durante a movimentação, ao aplicar vários filtros e outras propriedades. A interpolação de movimento pode ser criada interativamente com o Flash Professional, que gera o seu respectivo código ActionScript. No Flash, use Copiar movimento como o comando do ActionScript 3.0 para copiar o código que gerou a interpolação de movimento. Em seguida, será possível reutilizar o código ActionScript para gerar movimento em sua própria animação dinâmica no tempo de execução.

Consulte a seção Interpolações de movimento em *Uso do Flash Professional* para obter informações sobre a criação de interpolações de movimento.

Conceitos e termos importantes

A seguir há um termo importante que é relevante para este recurso:

Interpolação de movimento Construção que gera quadros intermediários de um objeto de exibição em diferentes estados e momentos, criando o efeito de que o primeiro estado evolui para o segundo estado suavemente. Ela é usada para movimentar um objeto de exibição no palco, além de fazê-lo aumentar, diminuir, rotacionar, atenuar ou mudar de cor no decorrer do tempo.

Cópia de scripts de interpolações de movimento no Flash

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

Uma interpolação gera quadros intermediários que mostram um objeto de exibição em diferentes estados em dois quadros diferentes de uma linha do tempo. Ela cria o efeito de que a imagem do primeiro quadro evolui suavemente até se tornar a imagem do segundo. Em uma interpolação de movimento, a mudança de aspecto geralmente envolve a mudança de posição do objeto de exibição, criando assim o movimento. Além do reposicionamento do objeto de exibição, uma interpolação de movimento também pode fazê-lo girar, inclinar, redimensionar ou ainda aplicar filtros a ele.

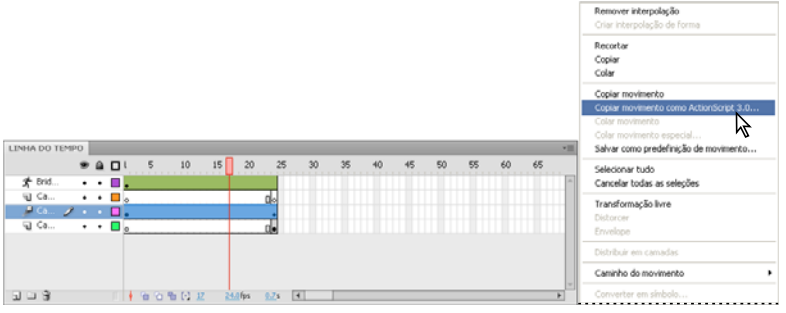
Crie uma interpolação de movimento no Flash, movendo um objeto de exibição entre os quadros-chave da linha do tempo. O Flash gera automaticamente o código ActionScript que descreve a interpolação, que pode ser copiado e salvo em um arquivo. Consulte a seção Interpolações de movimento em *Uso do Flash Professional* para obter informações sobre a criação de interpolações de movimento.

Você pode acessar Copiar movimento como um comando do ActionScript 3.0 no Flash de duas maneiras. A primeira é a partir de um menu de contexto de interpolação no palco:

1 Selecione a interpolação de movimento no palco.

2 Clique sobre ela com o botão direito (Windows) ou mantenha a tecla Ctrl pressionada (Macintosh).

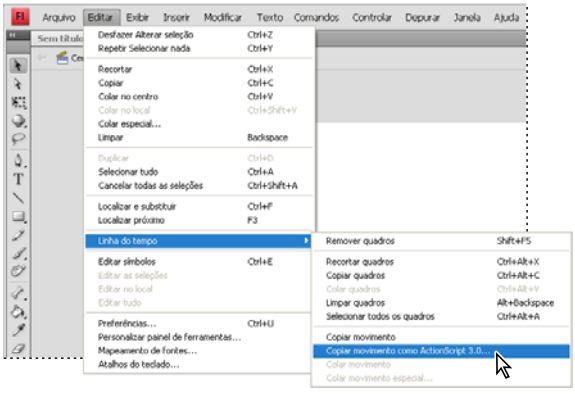
3 Selecione Copiar movimento como o comando do ActionScript 3.0. . .



A segunda maneira é selecionar o comando diretamente no menu Editar do Flash:

1 Selecione a interpolação de movimento no palco.

2 Selecione Editar > Linha do tempo > Copiar movimento como ActionScript 3.0.



Após copiar o script, cole-o em um arquivo e salve-o.

Após criar uma interpolação de movimento e copiar e salvar o script, você pode reutilizá-lo ou modificá-lo na animação dinâmica com base em ActionScript que criou.

Incorporação de scripts de interpolações de movimento

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

O cabeçalho do código ActionScript copiado do Flash lista todos os módulos necessários ao suporte da interpolação de movimento.

Classes de interpolação de movimento

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

As classes essenciais de interpolação de movimento são AnimatorFactory, MotionBase e as classes Motion contidas no pacote `fl.motion`. Talvez você precise de classes adicionais, dependendo das propriedades manipuladas pela interpolação de movimento. Por exemplo, se a interpolação de movimento transforma ou rotaciona o objeto de exibição, importe as classes `flash.geom` adequadas. Se ela aplicar filtros, importe as classes `flash.filter`. No ActionScript, uma interpolação de movimento é uma ocorrência da classe Motion. A classe Motion armazena uma sequência de animação de quadros-chave que pode ser aplicada a um objeto visual. Os dados da animação abrangem posição, escala, rotação, inclinação, cor, filtros e atenuação.

O código ActionScript a seguir foi copiado de uma interpolação de movimento criada no Flash para animar um objeto de exibição cujo nome de ocorrência é `Symbol1_2`. Ele declara uma variável para um objeto MotionBase chamado `__motion_Symbol1_2`. A classe MotionBase é pai da classe Motion.

```
var __motion_Symbol1_2:MotionBase;
```

Em seguida, o script cria o objeto Motion:

```
__motion_Symbol1_2 = new Motion();
```

nomes de objetos Motion

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

No caso anterior, o Flash gerou automaticamente o nome `__motion_Symbol1_2` para o objeto Motion. O prefixo `__motion_` é adicionado ao nome do objeto de exibição. Desse modo, o nome gerado automaticamente é baseado no nome de ocorrência do objeto de destino da interpolação de movimento no Flash. A propriedade `duration` do objeto Motion indica o número total de quadros na interpolação de movimento:

```
__motion_Symbol1_2.duration = 200;
```

Por padrão, o Flash nomeia automaticamente a ocorrência do objeto de exibição cuja interpolação de movimento está sendo copiada, caso ainda não tenha um nome de instância.

Ao reutilizar o ActionScript criado pelo Flash em sua própria animação, você pode manter o nome que o Flash gera automaticamente para a interpolação ou você pode substituir um nome diferente. Se o nome da interpolação for alterado, certifique-se de alterá-lo em todo o script.

Opcionalmente, é possível atribuir um nome de sua preferência ao objeto de destino da interpolação de movimento no Flash. Em seguida, crie a interpolação de movimento e copie o script. Independentemente do nome que utilizar, certifique-se de que cada objeto Motion de seu código ActionScript tenha um nome exclusivo.

Descrição da animação

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

O método `addPropertyArray()` da classe MotionBase adiciona uma matriz de valores para descrever cada propriedade interpolada.

Potencialmente, a matriz contém um item para cada quadro-chave na interpolação de movimento. Em geral, algumas dessas matrizes contêm uma quantidade de itens menor do que o número total de quadros-chave na interpolação de movimento. Essa situação ocorre quando o último valor da matriz não é alterado nos quadros restantes.

Se o comprimento do argumento da matriz for maior do que a propriedade `duration` do objeto Motion, `addPropertyArray()` fará o ajuste do valor da propriedade `duration` de modo adequado. Ela não adiciona quadros-chave às propriedades que foram adicionadas anteriormente. Os quadros-chave recém-adicionados permanecem nos quadros extras da animação.

As propriedades `x` e `y` do objeto Motion descrevem a mudança de posição do objeto interpolado no decorrer da execução da animação. Essas coordenadas são os valores que apresentam maior possibilidade de mudança em cada quadro-chave, se a posição do objeto de exibição for alterada. Você pode adicionar propriedades de movimento adicionais com o método `addPropertyArray()`. Por exemplo, adicione os valores `scaleX` e `scaleY` se o objeto interpolado estiver redimensionado. Adicione os valores `cewX` e `skewY` se ele estiver inclinado. Adicione a propriedade `rotationConcat` se ele estiver em rotação.

Use o método `addPropertyArray()` para definir as seguintes propriedades de interpolação:

`x`	Posição horizontal do ponto de transformação do objeto no espaço coordenado de seu objeto pai.
`y`	Posição vertical do ponto de transformação do objeto no espaço coordenado de seu objeto pai.
`z`	Posição de profundidade (eixo z) do ponto de transformação do objeto no espaço coordenado de seu objeto pai.
`scaleX`	Escala horizontal como uma porcentagem do objeto, como aplicada a partir do ponto de transformação.

`scaleY`	Escala vertical como porcentagem do objeto, como aplicada a partir do ponto de transformação.
`skewX`	Ângulo de inclinação horizontal do objeto em graus, como aplicado a partir do ponto de transformação.
`skewY`	Ângulo de inclinação vertical do objeto em graus, como aplicado a partir do ponto de transformação.
`rotationX`	Rotação do objeto ao redor do eixo x, a partir de sua orientação original.
`rotationY`	Rotação do objeto ao redor do eixo y, a partir de sua orientação original.
`rotationConcat`	Valores de rotação (eixo z) do objeto no movimento relacionado à orientação anterior, como aplicados a partir do ponto de transformação.
`useRotationConcat`	Se definido, faz com que o objeto de destino gire quando `addPropertyArray()` fornecer dados de movimento.
`blendMode`	Valor da classe BlendMode especificando a mistura de cores de um objeto com gráficos abaixo.
`matrix3D`	Propriedade matrix3D, caso haja alguma no quadro-chave (usada em interpolações em 3D). Se usada, todas as propriedades de transformação anteriores serão ignoradas.
`rotationZ`	Rotação do eixo z do objeto, em graus, a partir da sua orientação original referente ao contêiner 3D pai. Usado em interpolações em 3D no lugar de rotationConcat.

As propriedades adicionadas ao script gerado automaticamente dependem das propriedades que foram atribuídas à interpolação de movimento no Flash. Você pode adicionar, remover ou modificar algumas dessas propriedades ao personalizar a sua versão do script.

O código a seguir atribui valores às propriedades de uma interpolação de movimento chamada `__motion_Wheel`. Nesse caso, o objeto de exibição interpolado não muda de lugar, mas gira na mesma posição nos 29 quadros da interpolação de movimento. Os valores múltiplos atribuídos à matriz `rotationConcat` definem a rotação. Os outros valores de propriedades dessa interpolação de movimento não são alterados.

```
__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"]);
```

No exemplo a seguir, o objeto de exibição chamado `Leaf_1` se move no palco. Suas matrizes de propriedade `x` e `y` contêm valores diferentes para cada um dos 100 quadros da animação. Além disso, o objeto gira em torno de seu eixo `z` ao mesmo tempo em que se move pelo palco. Os itens múltiplos da matriz de propriedade `rotationZ` determinam a rotação.

```
__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,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ]
);
__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"]);
```

Adição de filtros

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

Se o objeto de destino de uma interpolação de movimento contiver filtros, esses filtros serão adicionados usando os métodos `initFilters()` e `addFilterPropertyArray()` da classe Motion.

Inicialização da matriz de filtros

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

O método `initFilters()` inicializa os filtros. O seu primeiro argumento é uma matriz dos nomes de classe totalmente qualificados de todos os filtros aplicados ao objeto de exibição. Essa matriz de nomes de filtros é gerada a partir da lista de filtros da interpolação de movimento no Flash. Em sua cópia do script, é possível remover ou adicionar quaisquer filtros no pacote `flash.filters` dessa ou a essa matriz. A chamada a seguir inicializa a lista de filtros para o objeto de exibição de destino. Ela aplica os filtros `DropShadowFilter`, `GlowFilter` e `BevelFilter` e copia a lista para cada quadro-chave no objeto Motion.

```
__motion_Box.initFilters(["flash.filters.DropShadowFilter", "flash.filters.GlowFilter",
"flash.filters.BevelFilter"], [0, 0, 0]);
```

Adição de filtros

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

O método `addFilterPropertyArray()` descreve as propriedades de um filtro inicializado com os seguintes argumentos:

1. O primeiro argumento identifica um filtro por índice. O índice refere-se à posição do nome do filtro na matriz de nomes de classes de filtro transferidas em uma chamada anterior de `initFilters()`.
2. O segundo argumento é a propriedade de filtro para armazenamento desse filtro em cada quadro-chave.
3. O terceiro argumento é o valor da propriedade de filtro especificada.

Dada a chamada anterior de `initFilters()`, as chamadas seguintes de `addFilterPropertyArray()` atribuem um valor 5 às propriedades `blurX` e `blurY` de `DropShadowFilter`. `DropShadowFilter` é o primeiro (índice 0) item da matriz dos filtros inicializados:

```
__motion_Box.addFilterPropertyArray(0, "blurX", [5]);
__motion_Box.addFilterPropertyArray(0, "blurY", [5]);
```

As três chamadas a seguir atribuem valores às propriedades quality, alpha e color de `GlowFilter`, o segundo item (índice 1) da matriz do filtro inicializado:

```
__motion_Box.addFilterPropertyArray(1, "quality", [BitmapFilterQuality.LOW]);
__motion_Box.addFilterPropertyArray(1, "alpha", [1.00]);
__motion_Box.addFilterPropertyArray(1, "color", [0xff0000]);
```

As quatro chamadas a seguir atribuem valores a `shadowAlpha`, `shadowColor`, `highlightAlpha` e `highlightColor` de `BevelFilter`, o terceiro (índice 2) item da matriz de filtros inicializados:

```
__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]);
```

Ajuste da cor com ColorMatrixFilter

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

Após a inicialização de `ColorMatrixFilter`, você poderá definir as propriedades `AdjustColor` adequadas para ajustar o brilho, o contraste, a saturação e a matiz do objeto de exibição interpolado. Em geral, o filtro `AdjustColor` é aplicado quando a interpolação de movimento é criada no Flash; você pode refiná-la em sua cópia do ActionScript. O exemplo a seguir transforma a matiz e a saturação do objeto de exibição à medida que ele se movimenta.

```
__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);
```

Associação de uma interpolação de movimento com seus objetos de exibição

O Flash Player 9 e superior, Adobe AIR 1.0 e superior, requerem o Flash CS3 ou superior

A última tarefa é associar a interpolação de movimento ao objeto de exibição ou aos objetos que ele manipula.

A classe AnimatorFactory gerencia a associação entre uma interpolação de movimento e seus objetos de exibição de destino. O argumento para o construtor AnimatorFactory é o objeto Motion:

```
var __animFactory_Wheel:AnimatorFactory = new AnimatorFactory(__motion_Wheel);
```

Use o método `addTarget()` da classe AnimatorFactory para associar o objeto de exibição de destino à sua interpolação de movimento. O ActionScript copiado do Flash gera um comentário para a linha `addTarget()` e não especifica um nome de ocorrência:

```
// __animFactory_Wheel.addTarget(<instance name goes here>, 0);
```

Em sua cópia, especifique o objeto de exibição para associá-lo à interpolação de movimento. No exemplo a seguir, os destinos são especificados como `greenWheel` e `redWheel`:

```
__animFactory_Wheel.AnimatorFactory.addTarget(greenWheel, 0);
__animFactory_Wheel.AnimationFactory.addTarget(redWheel, 0);
```

Você pode associar vários objetos de exibição com a mesma interpolação de movimento usando várias chamadas de `addTarget()`.

Capítulo 18: Trabalho com cinemática inversa

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

A IK (Cinemática inversa) é uma ótima técnica de criação de movimentos realistas.

A IK permite a criação de movimentos coordenados dentro de uma cadeia de trechos conectados conhecida como armadura IK, para que as partes apresentem movimentos sincronizados realistas. As partes da armadura são os bones e as junções. Dado o ponto final da armadura, a IK calcula os ângulos das junções que devem alcançar esse ponto.

Fazer os cálculos de tais ângulos manualmente seria desafiador. O mérito desse recurso é que você pode criar armaduras interativamente usando o Adobe® Flash® Professional. Em seguida, é só animá-las usando o ActionScript. O mecanismo IK incluído com o Flash Professional realiza os cálculos para descrever o movimento da armadura. É possível limitar o movimento a determinados parâmetros contidos em seu código ActionScript.

Uma novidade do Flash Professional CS5 da versão IK é o conceito de bone spring, normalmente associado com aplicativos de animação de ponta. Utilizado com o Physics Engine, este recursos permite configurar movimentos realísticos. Este efeito é visível no tempo de execução e durante a autoria.

Para criar armaduras de cinemática inversa, é preciso ter a licença do Flash Professional.

Mais tópicos da Ajuda

Pacote fl.ik

Noções básicas de cinemática inversa

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

A IK permite a criação de animações realistas por meio da junção de partes que se movimentam entre si.

Por exemplo, com o uso da IK você pode mover uma perna para uma determinada posição ao articular o movimento de suas respectivas junções para alcançar a posição desejada. A IK usa uma estrutura de bones encadeados em uma estrutura chamada armadura IK. O pacote `fl.ik` ajuda a criar animações que remetem a movimentos naturais. Ele permite que você anime várias armaduras IK diretamente sem precisar saber muito sobre o funcionamento dos algoritmos IK.

Crie a armadura IK com seus respectivos bones e junções auxiliares no Flash Professional. Em seguida, acesse as classes IK para animá-las no tempo de execução.

Consulte a seção Uso de cinemática inversa em *Uso do Flash Professional* para obter instruções detalhadas sobre como criar uma armadura IK.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que são relevantes a este recurso:

Armadura Cadeia cinemática, formada por bones e junções, utilizada em animações computadorizadas para simular movimentos reais.

Bone é um segmento rígido de uma armadura, que corresponde a um dos ossos de um esqueleto.

Cinemática inversa (IK) Processo de determinação dos parâmetros de um objeto com junções flexíveis, que pode ser uma cadeia cinemática ou uma armadura.

Junção Local em que ocorre a união de dois bones, construída para permitir o movimento dos bones, que corresponde a uma articulação de um ser vivo.

Physics Engine Um pacote de algoritmos de Física usados para criar ações realistas em animações.

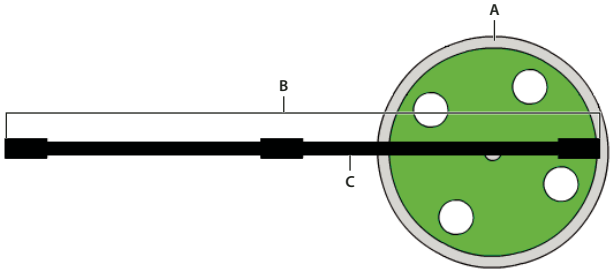
Spring A qualidade de um osso que se move e reage quando o osso ascendente é movimentado e, depois, diminui gradualmente ao longo do tempo.

Visão geral da animação de armaduras IK

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

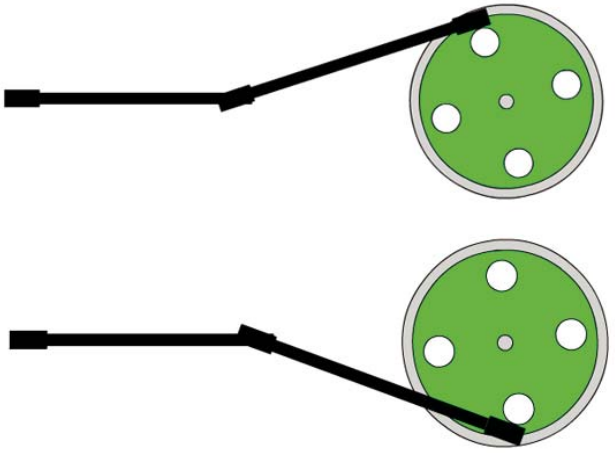
Após criar uma armadura IK no Flash Professional, use as classes `fl.ik` para restringir seus movimentos, rastrear seus eventos e animá-la no tempo de execução.

A figura a seguir apresenta um clipe de filme chamado `Roda`. O eixo é uma ocorrência de uma IKArmature chamada `Eixo`. A classe IKMover move a armadura de modo sincronizado com a rotação da roda. O IKBone, `ikBone2`, da armadura é anexado à roda, como junção de sua parte inferior.



A. *Roda* ***B.*** *Eixo* ***C.*** *ikBone2*

No tempo de execução, a roda gira em associação com a interpolação de movimento `__motion_Wheel` discutida em "Descrição da animação" na página 347. Um objeto IKMover inicia e controla o movimento do eixo. A figura a seguir exibe dois instantâneos da armadura do eixo fixos à roda em diferentes quadros durante o movimento de rotação.



No tempo de execução, o ActionScript a seguir:

- Obtém informações sobre a armadura e seus componentes
- Cria ocorrência de um objeto IKMover
- Move o eixo de acordo com a rotação da roda

```
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);
    }
}
```

As classes IK usadas para mover o eixo são:

- IKArmature: descreve a armadura, uma estrutura em árvore formada por bones e junções, que deve ser criada no Flash Professional
- IKManager: classe de contêiner para todas as armaduras IK do documento, que deve ser criada no Flash Professional
- IKBone: um segmento de uma armadura IK
- IKJoint: uma conexão entre dois bones IK
- IKMover: inicia e controla o movimento IK das armaduras

Veja descrições completas e detalhadas dessas classes no pacote ik.

Obtenção de informações sobre uma armadura IK

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

Primeiro, declare variáveis para a armadura, para o bone e para a junção que une as partes que deseja movimentar.

O código a seguir usa o método `getArmatureByName()` da classe IKManager para atribuir o valor da armadura Eixo à variável `tree` da IKArmature. A armadura Axle foi criada anteriormente no Flash Professional.

```
var tree:IKArmature = IKManager.getArmatureByName("Axle");
```

De modo semelhante, o código a seguir usa o método `getBoneByName()` da classe IKArmature para atribuir a variável IKBone ao valor do bone `ikBone2`.

```
var bone:IKBone = tree.getBoneByName("ikBone2");
```

A junção da parte inferior do bone `ikBone2` é a parte da armadura que a conecta à roda.

A linha a seguir declara a variável `endEffector` e atribui a ela a propriedade `tailjoint` do bone `ikBone2`:

```
var endEffector:IKJoint = home.tailjoint;
```

A variável `pos` é um ponto que armazena a posição correta da junção `endEffector`.

```
var pos:Point = endEffector.position;
```

Nesse exemplo, `pos` é a posição da junção na extremidade do eixo conectada à roda. O valor original dessa variável é obtida a partir da propriedade `position` do IKJoint.

Ocorrência de IKMover e limitação de seu movimento

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

Uma ocorrência da classe IKMover movimenta o eixo.

A linha a seguir cria ocorrência do objeto `ik` do IKMover, transmitindo ao seu construtor o elemento a ser movimentado e o ponto de início do movimento:

```
var ik:IKMover = new IKMover(endEffector, pos);
```

As propriedades da classe IKMover permitem a limitação do movimento de uma armadura. É possível limitar o movimento, tendo como base a distância, as iterações e o tempo de duração do movimento.

Os pares de propriedades a seguir reforçam esses limites. Os pares consistem em um valor booliano que indica se o movimento é limitado, além de um inteiro que especifica o limite:

Propriedade Boolean	Propriedade Integer	Limite definido
`limitByDistance:Boolean`	`distanceLimit:int`	Define a distância máxima em pixels que o mecanismo IK se desloca para cada iteração.
`limitByIteration:Boolean`	`iterationLimit:int`	Define o número máximo de iterações que o mecanismo IK executa para cada movimento.
`limitByTime:Boolean`	`timeLimit:int`	Define o tempo máximo em milissegundos designado para que o mecanismo IK execute o movimento.

Por padrão, todos os valores boolianos são definidos como `false`, assim o movimento não é limitado a menos que você defina explicitamente um valor booliano como `true`. Para forçar um limite, defina a propriedade apropriada `true` e depois especifique um valor para a propriedade Integer correspondente. Se você definir o limite para um valor sem configurar sua propriedade Boolean correspondente, o limite será ignorado. Nesse caso, o mecanismo IK continua a movimentação do objeto até outro limite ou até que a posição de destino do IKMover seja alcançada.

No exemplo a seguir, a distância máxima do movimento da armadura está definido como 0,1 pixels por iteração. O número máximo de iterações para cada movimento está definido como dez.

```
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;
```

Movimentação de uma armadura IK

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

O IKMover movimenta o eixo interno do ouvinte de eventos para a roda. Em cada evento enterFrame da roda, uma nova posição de destino para a armadura é calculada. Usando seu método `moveTo()` , o IKMover movimenta a junção da parte inferior para sua respectiva posição de destino ou o mais próximo possível, dentro dos limites definidos pelas suas propriedades `limitByDistance`, `limitByIteration` e `limitByTime`.

```
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);
    }
}
```

Utilizando suspensões

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS5 ou superior

A cinemática invertida o Flash Professional CS5 fornece suporte a bone spring. O bone spring pode ser configurado durante a autoria e seus atributos podem ser adicionados ou modificados em tempo de execução. Elasticidade (spring) é uma propriedade de um osso e de suas articulações. Ele possui dois atributos: `IKJoint.springStrength`, que define a quantidade de spring e `IKJoint.springDamping`, que adiciona resistência ao valor de força e altera a taxa de decaimento do spring.

A força do spring é um valor percentual de padrão 0 (totalmente rígido) até 100 (muito solto e controlado por propriedades físicas). Ossos com elasticidade reagem ao movimento de suas articulações. Se nenhuma outra translação (rotação, x, ou y) estiver ativada, as configurações de elasticidade não terão nenhum efeito.

O amortecimento de spring é um valor percentual de padrão 0 (nenhuma resistência) até 100 (muito reduzido). O amortecimento altera o tempo entre o movimento inicial o osso e o retorno à posição de descanso.

Verifique se os springs estão habilitados para um objeto IKArmature verificando a propriedade `IKArmature.springsEnabled` . As outras propriedades e os outros métodos da elasticidade pertencem a cada objeto IKJoint individualmente. Uma junta pode ser habilitada para rotação angular e transporte entre os eixo x e y. Você pode posicionar um ângulo de spring de junta com `IKJoint.setSpringAngle` e uma posição de junta de spring transportável com `IKJoint.setSpringPt`.

Este exemplo seleciona um osso pelo nome e identifica sua tailJoint. O código testa a armadura ascendente para saber se as elasticidades estão ativadas e, em seguida, define as propriedades de elasticidade da articulação.

```
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
    }
}
```

Uso de eventos IK

O Flash Player 10 e superior, Adobe AIR 1.5 e superior, requerem o Flash CS4 ou superior

A classe IKEvent permite a criação de um objeto de evento que contém informações sobre eventos IK. As informações IKEvent descrevem o movimento que foi interrompido porque o tempo especificado, a distância ou o limite de iteração foi excedido.

O código a seguir mostra um ouvinte de eventos e um manipulador para rastreamento de eventos de limite de tempo. Esse manipulador de eventos indica o tempo, a distância, a contagem de iterações e as propriedades de junções de um evento que dispara quando o limite de tempo do IKMover é excedido.

```
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);
}
```

Capítulo 19: Trabalho em 3D (três dimensões)

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O Flash Player e os tempos de execução do AIR suportam gráficos 3D de duas maneiras. Você pode usar objetos de exibição tridimensionais na lista de exibição do Flash. Isso é adequado na adição de efeitos tridimensionais em conteúdo do Flash e em objetos com baixa contagem de polígonos. No Flash Player 11 e no AIR 3 ou em versões posteriores você pode renderizar cenas 3D complexas usando a API do Stage3D.

Uma porta de visualização de Stage3D não é um objeto de exibição. Em vez disso, os gráficos 3D são renderizados para uma porta de visualização que é exibida embaixo da lista de exibição do Flash (e acima de quaisquer planos da porta de visualização de StageVideo). Em vez de usar as classes DisplayObject do Flash para criar uma cena, você pode usar um canal 3D programável (similar ao OpenGL e ao Direct3D). Esse canal recebe os dados de triângulo e as texturas como entrada e renderiza a cena usando os programas de sombreador fornecidos. A aceleração de hardware é usada quando uma unidade de processamento de imagem compatível (GPU) com os drivers suportados está disponível no computador do cliente.

O Stage3D fornece uma API de nível muito baixo. Em um aplicativo, você é incentivado a usar uma estrutura 3D que suporte o Stage3D. Você pode criar sua própria estrutura ou usar uma das várias estruturas comerciais e de código livre já disponíveis.

Para obter mais informações sobre o desenvolvimento de aplicativos 3D usando o Stage3D e sobre as estruturas 3D disponíveis, visite o Centro do Desenvolvedor do Flash Player: Stage 3D.

Noções básicas de objetos de exibição 3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A principal diferença entre um objeto bidimensional (2D) e um objeto tridimensional (3D) projetados em uma tela bidimensional é a adição de uma terceira dimensão ao objeto. A terceira dimensão permite que o objeto se aproxime ou se afaste do ponto de visão do usuário.

Quando você define explicitamente a propriedade `z` de um objeto de exibição com um valor numérico, o objeto automaticamente cria uma matriz de transformação 3D. É possível alterar essa matriz para modificar as configurações de transformação 3D do objeto.

Além disso, a rotação 3D é diferente da rotação 2D. Em 2D, o eixo de rotação está sempre perpendicular ao plano x/y - em outras palavras, ele está no eixo z. Em 3D, o eixo de rotação pode estar em torno de qualquer um dos eixos x, y ou z. Definir as propriedades de rotação e dimensionamento de um objeto de exibição permite que ele se mova no espaço 3D.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você vai encontrar ao conhecer o gráfico de programação em três dimensões:

Perspectiva Em um plano 2D, a representação de linhas paralelas como convergentes em um ponto de fuga para dar a ilusão de profundidade e distância.

Projeção A produção de uma imagem 2D de um objeto de dimensões maiores; a projeção 3D mapeia pontos 3D para um plano 2D.

Rotação Alterar a orientação (e, muitas vezes, a posição) de um objeto movendo cada ponto incluído nele em movimento circular.

Transformação Alterar pontos 3D ou conjuntos de pontos por translação, rotação, escala, inclinação ou uma combinação destas ações.

Translação Alterar a posição de um objeto movendo cada ponto incluído nele da mesma forma e na mesma direção.

Ponto de fuga Ponto em que linhas paralelas recuadas parecem se encontrar quando representadas em perspectiva linear.

Vetor Um vetor 3D representa um ponto ou um local no espaço tridimensional com as coordenadas cartesianas x, y e z.

Vértice Um ponto de canto.

Malha com textura Qualquer ponto que define um objeto no espaço 3D.

Mapeamento UV Uma forma de aplicar textura ou bitmap a uma superfície 3D. O mapeamento UV atribui valores a coordenadas de uma imagem como porcentagens dos eixos horizontal (U) e vertical (V).

valor T O fator de dimensionamento para determinar o tamanho de um objeto 3D à medida que ele se aproxima, ou se afasta, do ponto de visão atual.

Remoção Renderização, ou não, de superfícies com contorno específico. O uso da remoção pode ocultar superfícies que não estão visíveis para o ponto de visão atual.

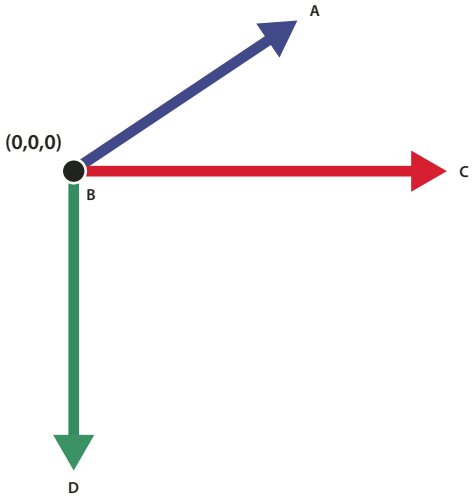
Noções básicas de objetos de exibição 3D do Flash Player e de tempo de execução do AIR

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Nas versões do Flash Player anteriores ao Flash Player 10 e nas versões do Adobe AIR anteriores ao Adobe AIR 1.5, os objetos de exibição têm duas propriedades, `x` e `y`, para posicioná-los em um plano 2D. A partir do Flash Player 10 e do Adobe AIR 1.5, todo objeto de exibição do ActionScript tem uma propriedade `z` que permite posicioná-lo ao longo do eixo z, normalmente usado para indicar profundidade ou distância.

O Flash Player 10 e o Adobe AIR 1.5 introduziram o suporte para efeitos 3D. No entanto, os objetos de exibição são inerentemente planos. Cada objeto de exibição, como MovieClip ou Sprite, basicamente renderiza-se em duas dimensões em um único plano. Os recursos 3D permitem colocar, movimentar, girar e transformar todos esses objetos planos em tridimensionais. Eles também permitem gerenciar pontos 3D e convertê-los em coordenadas x, y 2D para que você possa projetar objetos 3D em uma exibição 2D. É possível simular muitos tipos de experiências 3D usando esses recursos.

O sistema de coordenadas 3D usado pelo ActionScript difere do de outros sistemas. Quando são usadas coordenadas 2D no ActionScript, o valor de x aumenta à medida que você vai para a direita no eixo x, e o valor de y aumenta conforme você percorre o eixo y. O sistema de coordenadas 3D retém essas convenções e adiciona um eixo z cujo valor aumenta à medida que você se afasta do ponto de visão.



As direções positivas dos eixos x, y e z no sistema de coordenadas 3D do ActionScript.
A. *Eixo +Z* ***B.*** *Origem* ***C.*** *Eixo +X* ***D.*** *Eixo +Y*

Nota: *O Flash Player e o AIR sempre representam 3D em camadas. Isso significa que, se o objeto A está na frente do objeto B na lista de exibição, o Flash Player ou o AIR sempre renderiza A na frente de B, independentemente dos valores do eixo z dos dois objetos. Para resolver este conflito entre a ordem na lista de exibição e a ordem no eixo z, use o método* `transform.getRelativeMatrix3D()` *para salvar e reordenar as camadas de objetos de exibição 3D. Para obter mais informações, consulte "Uso de objetos Matrix3D para reordenar a exibição" na página 372.*

As seguintes classes do ActionScript dão suporte aos novos recursos relacionados a 3D:

1. A classe flash.display.DisplayObject contém a propriedade `z` e novas propriedades de rotação e dimensionamento para manipular objetos de exibição no espaço 3D. O método `DisplayObject.local3DToGlobal()` oferece uma maneira simples de projetar geometria 3D em um plano 2D.
2. A classe flash.geom.Vector3D pode ser usada como estrutura de dados para gerenciar pontos 3D. Ela também oferece suporte para matemática de vetores.
3. A classe flash.geom.Matrix3D dá suporte a transformações complexas de geometria 3D, como rotação, dimensionamento e translação.
4. A classe flash.geom.PerspectiveProjection controla os parâmetros para mapear geometria 3D em uma exibição 2D.

Existem duas abordagens distintas para simular imagens 3D no ActionScript:

1. Organizar e animar objetos planos no espaço 3D. Esta abordagem envolve animar objetos de exibição usando as propriedades x, y e z de objetos de exibição ou definir propriedades de rotação e dimensionamento utilizando a classe DisplayObject. É possível obter movimentos mais complexos usando o objeto DisplayObject.transform.matrix3D. O objeto DisplayObject.transform.perspectiveProjection personaliza a forma como os objetos de exibição são desenhados em perspectiva 3D. Use esta abordagem quando quiser animar objetos 3D formados principalmente por planos. Exemplos desta abordagem incluem galerias de imagens 3D ou objetos de animação 2D organizados no espaço 3D.

2 Gerar triângulos 2D a partir de geometria 3D e renderizá-los com texturas. Para usar esta abordagem, primeiro você deve definir e gerenciar dados sobre objetos 3D e, em seguida, converter esses dados em triângulos 2D para fins de renderização. É possível mapear texturas de bitmap para esses triângulos, e depois eles são desenhados em um objeto gráfico através do método `Graphics.drawTriangles()`. Exemplos desta abordagem incluem carregar dados de modelo 3D a partir de um arquivo e renderizar o modelo na tela ou gerar e desenhar um terreno 3D como malhas de triângulo.

Criação e movimentação de objetos de exibição 3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Para converter um objeto de exibição 2D em um objeto de exibição 3D, você pode definir explicitamente a propriedade `z` correspondente com um valor numérico. Quando você atribui um valor para a propriedade `z`, é criado um novo objeto Transform para o objeto de exibição. Definir as propriedades `DisplayObject.rotationX` ou `DisplayObject.rotationY` também cria um novo objeto Transform. O objeto Transform contém uma propriedade `Matrix3D` que determina como o objeto de exibição é representado no espaço 3D.

O seguinte código define as coordenadas para um objeto de exibição chamado “leaf” (folha):

```
leaf.x = 100; leaf.y = 50; leaf.z = -30;
```

Você pode ver estes valores, bem como as propriedades derivadas deles, na propriedade `matrix3D` do objeto Transform da folha:

```
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);
```

Para obter informações sobre as propriedades do objeto Transform, consulte a classe Transform. Para obter informações sobre as propriedades do objeto Matrix3D, consulte a classe Matrix3D.

Movimentação de um objeto no espaço 3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível movimentar um objeto no espaço 3D alterando os valores de suas propriedades `x, y ou z`. Quando você altera o valor da propriedade `z`, o objeto parece aproximar-se ou afastar-se do visualizador.

O código a seguir movimenta duas elipses para frente e para trás ao longo dos eixos z alterando o valor das respectivas propriedades `z` em resposta a um evento. `ellipse2` movimenta-se mais rápido do que `ellipse1`: sua propriedade `z` é aumentada por um múltiplo de 20 em cada evento Frame, enquanto a propriedade `z` de `ellipse1` é aumentada por um múltiplo de 10:

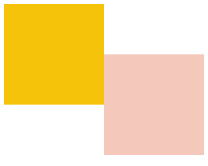
```
var depth:int = 1000;

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;
    }
}
```

Rotação de um objeto no espaço 3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível girar um objeto de três formas diferentes, dependendo de como você definir as propriedades de rotação do objeto: `rotationX`, `rotationY` e `rotationZ`.

A figura abaixo mostra dois quadrados que não são girados:

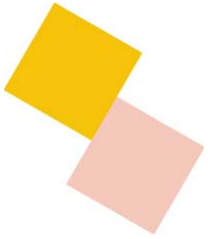


A próxima figura mostra os dois quadrados quando você incrementa a propriedade `rotationY` do contêiner dos quadrados para girá-los no eixo y. Girar o contêiner, ou o objeto de exibição pai, dos dois quadrados gira ambos os quadrados:

```
container.rotationY += 10;
```

A próxima figura mostra o que acontece quando você define a propriedade `rotationX` do contêiner dos quadrados. Isso gira os quadrados no eixo x.

A próxima figura mostra o que acontece quando você incrementa a propriedade `rotationZ` do contêiner dos quadrados. Isso gira os quadrados no eixo z.



Um objeto de exibição pode, ao mesmo tempo, se movimentar e girar no espaço 3D.

Projeção de objetos 3D em uma exibição 2D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A classe PerspectiveProjection do pacote `flash.geom` oferece uma maneira simples de aplicar perspectiva rudimentar quando você movimenta objetos de exibição pelo espaço 3D.

Caso você não crie uma projeção em perspectiva para o espaço 3D explicitamente, o mecanismo 3D usará um objeto PerspectiveProjection padrão que existe na raiz e é propagado para todos os seus filhos.

As três propriedades que definem como um objeto PerspectiveProjection exibe o espaço 3D são:

- `fieldOfView`
- `projectionCenter`
- `focalLength`

Modificar o valor de `fieldOfView` modifica automaticamente o valor de `focalLength` e vice-versa, uma vez que elas são interdependentes.

A fórmula usada para calcular a propriedade `focalLength` considerando-se o valor de `fieldOfView` é a seguinte:

```
focalLength = stageWidth/2 * (cos(fieldOfView/2) / sin(fieldOfView/2)
```

Normalmente você modificaria a propriedade `fieldOfView` de maneira explícita.

Campo de visão

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Manipulando a propriedade `fieldOfView` da classe PerspectiveProjection, é possível fazer com que um objeto de exibição 3D que está se aproximando do visualizador pareça maior e com que um objeto que está se afastando do visualizador pareça menor.

A propriedade `fieldOfView` especifica um ângulo entre 0 e 180 graus que determina a intensidade da projeção em perspectiva. Quanto maior o valor, mais intensa a distorção aplicada a um objeto de exibição que se movimenta em seu eixo z. Um valor baixo de `fieldOfView` resulta em um dimensionamento muito pequeno e faz com que os objetos pareçam se movimentar apenas um pouco para trás no espaço. Um valor alto de `fieldOfView` gera mais distorção e o aspecto de maior movimento. O valor máximo de 179,9999... graus resulta em um efeito extremo da lente de olho de peixe. O valor máximo de `fieldOfView` é 179,9999... e o mínimo é 0,00001... 0 e 180 exatos são valores inválidos.

Centro da projeção

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `projectionCenter` representa o ponto de fuga na projeção em perspectiva. Ela é aplicada como um deslocamento até o ponto de registro padrão (0,0) no canto superior esquerdo do palco.

Conforme um objeto parece se afastar do visualizador, ele se movimenta em direção ao ponto de fuga e, por fim, desaparece. Imagine um corredor infinitamente longo. Quando você olha para o final dele, as extremidades das paredes se convergem em um ponto de fuga no final do corredor.

Se o ponto de fuga está no centro do palco, o corredor desaparece em direção a um ponto no centro. O valor padrão da propriedade `projectionCenter` é o centro do palco. Se, por exemplo, você quiser que os elementos apareçam no lado esquerdo do palco e que uma área 3D apareça no lado direito, defina `projectionCenter` como um ponto no lado direito do palco para torná-lo o ponto de fuga da sua área de exibição 3D.

Distância focal

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `focalLength` representa a distância entre a origem do ponto de visão (0,0,0) e a localização do objeto de exibição em seu eixo z.

Uma distância focal longa é como uma teleobjetiva com uma visualização limitada e distâncias reduzidas entre objetos. Uma distância focal curta é como uma lente com grande abertura angular, com a qual você tem uma visualização ampla com muita distorção. Uma distância focal média assemelha-se ao que os olhos humanos veem.

Normalmente, a propriedade `focalLength` é recalculada dinamicamente durante a transformação de perspectiva à medida que o objeto de exibição se movimenta, mas é possível defini-la de forma explícita.

Valores padrão da projeção em perspectiva

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O objeto PerspectiveProjection padrão criado na raiz tem os seguintes valores:

- `fieldOfView: 55`
- `perspectiveCenter: stagewidth/2, stageHeight/2`
- `focalLength: stageWidth/ 2 * ( cos(fieldOfView/2) / sin(fieldOfView/2) )`

Esses são os valores que serão utilizados se você não criar seu próprio objeto PerspectiveProjection.

É possível instanciar seu próprio objeto PerspectiveProjection com a intenção de você mesmo modificar `projectionCenter` e `fieldOfView`. Nesse caso, os valores padrão do objeto recém-criado são os seguintes, de acordo com um tamanho de palco padrão de 500 por 500:

- `fieldOfView: 55`
- `perspectiveCenter: 250,250`
- `focalLength: 480.24554443359375`

Exemplo: Projeção em perspectiva

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O exemplo a seguir demonstra o uso da projeção em perspectiva para criar o espaço 3D. Ele mostra como você pode modificar o ponto de fuga e alterar a projeção em perspectiva do espaço através da propriedade `projectionCenter`. Essa modificação força o recálculo das propriedades `focalLength` e `fieldOfView` com a distorção concomitante do espaço 3D.

Este exemplo:

1. Cria uma entidade gráfica chamada `center`, como um círculo com cruzes
2. Atribui as coordenadas da entidade gráfica `center` à propriedade `projectionCenter` da propriedade`perspectiveProjection` da propriedade `transform` da raiz
3. Adiciona ouvintes de eventos para eventos de mouse que chamam manipuladores que modificam a propriedade `projectionCenter` para que ela siga a localização do objeto `center`
4. Cria quatro caixas com o estilo acordeão que formam as paredes do espaço em perspectiva

Quando você testar este exemplo, ProjectionDragger.swf, arraste o círculo para diferentes locais. O ponto de fuga segue o círculo, parando no lugar em que você soltá-lo. Observe as caixas que circundam o espaço se alongam e se tornam distorcidas quando você afasta o centro de projeção do centro do palco.

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo ProjectionDragger estão na pasta 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;
        }
    }
}
```

Para projeção em perspectiva mais complexa, use a classe Matrix3D.

Execução de transformações 3D complexas

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A classe Matrix3D permite transformar pontos 3D dentro de um espaço de coordenadas ou mapear pontos 3D de um espaço de coordenadas para outro.

Você não precisa conhecer a matemática de matrizes para usar a classe Matrix3D. A maior parte das operações de transformação comuns pode ser feita usando-se os métodos da classe. Você não precisa se preocupar em definir ou calcular explicitamente os valores de cada elemento da matriz.

Depois de definir a propriedade `z` de um objeto de exibição como um valor numérico, você pode recuperar a matriz de transformação dele usando a propriedade Matrix3D do objeto Transform do objeto de exibição:

```
var leafMatrix:Matrix3D = this.transform.matrix3D;
```

Você pode usar os métodos do objeto Matrix3D para executar translação, rotação, dimensionamento e projeção em perspectiva no objeto de exibição.

Use a classe Vector3D com suas propriedades `x`, `y` e `z` para gerenciar pontos 3D. Ela também pode representar um vetor espacial na física, que tem uma direção e uma magnitude. Os métodos da classe Vector3D permitem executar cálculos comuns com vetores espaciais, como cálculos de adição, produto escalar e produto complementar.

Nota: *A classe Vector3D não está relacionada à classe Vector do ActionScript. A classe Vector3D contém propriedades e métodos para definir e manipular pontos 3D, enquanto a classe Vector oferece suporte a matrizes de objetos com tipo.*

Criação de objetos Matrix3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Há três maneiras de criar ou recuperar objetos `Matrix3D`:

- Use o método do construtor `Matrix3D()` para instanciar uma nova matriz. O construtor `Matrix3D()` toma um objeto `Vector` que contém 16 valores numéricos e coloca cada valor em uma célula da matriz. Por exemplo:

  ```
  var rotateMatrix:Matrix3D = new Matrix3D(1,0,0,1, 0,1,0,1, 0,0,1,1, 0,0,0,1);
  ```

- Defina o valor da propriedade `z` de um objeto de exibição. Em seguida, recupere a matriz de transformação da propriedade `transform.matrix3D` desse objeto.
- Recupere o objeto Matrix3D que controla a exibição de objetos 3D no palco chamando o método `perspectiveProjection.toMatrix3D()` no objeto de exibição raiz.

Aplicação de várias transformações 3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível aplicar muitas transformações 3D de uma só vez usando um objeto Matrix3D. Por exemplo, para girar, dimensionar e, depois, movimentar um cubo, você pode aplicar três transformações separadas a cada ponta do cubo. No entanto, é muito mais eficiente pré-calcular várias transformações em um objeto Matrix3D e, em seguida, executar uma transformação de matriz em cada uma das pontas.

Nota: *A ordem em que as transformações de matriz são aplicadas é importante. Os cálculos de matriz não são comutativos. Por exemplo, o resultado de aplicar uma rotação seguida de uma translação é diferente do resultado de aplicar a mesma translação seguida da mesma rotação.*

O exemplo a seguir mostra duas maneiras de executar várias transformações 3D.

```
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;
        }
    }
}
```

No método `doTransforms()`, o primeiro bloco de código usa as propriedades DisplayObject para alterar a rotação, o dimensionamento e a posição de uma forma de retângulo. O segundo bloco de código usa os métodos da classe Matrix3D para fazer as mesmas transformações.

A principal vantagem de usar os métodos `Matrix3D` é que todos os cálculos são realizados primeiro na matriz. Em seguida, eles são aplicados ao objeto de exibição apenas uma vez, quando sua propriedade `transform.matrix3D` é definida. Configurar propriedades DisplayObject torna a leitura do código-fonte um pouco mais simples. Todavia, sempre que é definida uma propriedade de rotação ou dimensionamento, ela gera vários cálculos e altera diversas propriedades do objeto de exibição.

Se o seu código aplicará as mesmas transformações complexas a objetos de exibição mais de uma vez, salve o objeto Matrix3D como uma variável e, em seguida, aplique-o de novo repetidas vezes.

Uso de objetos Matrix3D para reordenar a exibição

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Conforme mencionado anteriormente, a ordem em camadas dos objetos de exibição na lista de exibição determina a ordem em camadas da exibição, independentemente dos eixos x relacionados. Se a sua animação transforma as propriedades de objetos de exibição em uma ordem diferente da ordem da lista de exibição, o visualizador poderá ver os objetos de exibição em uma disposição em camadas que não corresponde à disposição em camadas no eixo z. Por isso, um objeto que deve parecer mais distante do visualizador pode aparecer na frente de um objeto que está mais perto do visualizador.

Para assegurar que a disposição em camadas dos objetos de exibição 3D corresponda às profundidades relativas dos objetos, use uma abordagem como a seguinte:

1. Use o método `getRelativeMatrix3D()` do objeto Transform para obter os `eixos z` relacionados dos objetos de exibição 3D filho.
2. Use o método `removeChild()` para remover os objetos da lista de exibição.
3. Classifique os objetos de exibição com base nos valores do eixo z relacionado.
4. Use o método `addChild()` para adicionar os filhos de volta à lista de exibição na ordem inversa.

Essa reordenação assegura que seus objetos serão exibidos de acordo com os eixos z relacionados.

O código a seguir força a exibição correta das seis faces de uma caixa 3D. Ele reordena as faces da caixa depois que rotações foram aplicadas a ela:

```
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);
    }
}
```

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo estão na pasta Samples/ReorderByZ.

Uso de triângulos para obter efeitos 3D

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

No ActionScript, você executa transformações de bitmap usando o método `Graphics.drawTriangles()`, pois os modelos 3D são representados por um conjunto de triângulos no espaço. (Contudo, o Flash Player e o AIR não dão suporte a um buffer de profundidade, por isso os objetos de exibição são inerentemente planos, ou 2D. Isso é descrito em "Noções básicas de objetos de exibição 3D do Flash Player e de tempo de execução do AIR" na página 361.) O método `Graphics.drawTriangles()` é parecido com o método `Graphics.drawPath()` no sentido de que usa um conjunto de coordenadas para desenhar um caminho de triângulo.

Para se familiarizar com o uso de `Graphics.drawPath()`, consulte "Caminhos de desenho" na página 240.

O método `Graphics.drawTriangles()` usa um Vector.<Number> para especificar as localizações das pontas para o caminho do triângulo:

```
drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null, uvtData:Vector.<Number>
= null, culling:String = "none"):void
```

O primeiro parâmetro de `drawTriangles()` é o único necessário: o parâmetro `vertices`. Este parâmetro é um vetor de números que define as coordenadas através das quais os seus triângulos são desenhados. Cada três conjuntos de coordenadas (seis números) representa um caminho de triângulo. Sem o parâmetro `indices`, o comprimento do vetor deve sempre ser um fator de seis, uma vez que cada triângulo requer três pares de coordenadas (três conjuntos de dois valores x/y). Por exemplo:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([
        10,10,  100,10,  10,100,
        110,10, 110,100, 20,100]));
```

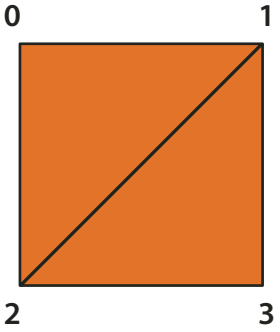
Nenhum desses triângulos compartilham pontas, mas, se compartilhassem, o segundo parâmetro de `drawTriangles()`, `indices`, poderia ser usado para reutilizar valores do vetor `vertices` para mais de um triângulo.

Quando usar o parâmetro `indices`, é importante que você saiba que os valores de `indices` são índices de pontos e não índices relacionados diretamente aos elementos de matriz `vertices`. Em outras palavras, um índice no vetor `vertices` conforme definido por `indices` é, na verdade, o índice real dividido por 2. Para a terceira ponta de um vetor `vertices`, por exemplo, use um valor de 2 para `indices`, mesmo que o primeiro valor numérico dessa ponta comece no índice de vetor de 4.

Por exemplo, mescle dois triângulos para que compartilhem a aresta diagonal usando o parâmetro `indices`:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([10,10, 100,10, 10,100, 100,100]),
    Vector.<int>([0,1,2, 1,3,2]));
```

Observe que, apesar de agora ter sido desenhado um quadrado usando-se dois triângulos, somente quatro pontas foram especificadas no vetor `vertices`. Usando `indices`, as duas pontas compartilhadas pelos dois triângulos são reutilizadas para cada triângulo. Isso diminui a contagem geral de vértices de 6 (12 números) para 4 (8 números):



Um quadrado desenhado com dois triângulos usando o parâmetro vertices

Esta técnica torna-se útil com malhas de triângulos maiores, onde a maioria das pontas são compartilhadas por vários triângulos.

Todos os preenchimentos podem ser aplicados a triângulos. Os preenchimentos são aplicados à malha de triângulos resultante, assim como ocorre com qualquer outra forma.

Transformação de bitmaps

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

As transformações de bitmap dão a ilusão de perspectiva ou "textura" em um objeto tridimensional. Especificamente, você pode distorcer um bitmap em direção a um ponto de fuga de modo que a imagem pareça diminuir à medida que se aproxima do ponto de fuga. Se preferir, você pode usar um bitmap bidimensional para criar uma superfície para um objeto tridimensional, dando a ilusão de textura ou “wrapping” nesse objeto 3D.

Uma superfície bidimensional usando um ponto de fuga e um objeto tridimensional delimitado com um bitmap.

Mapeamento UV

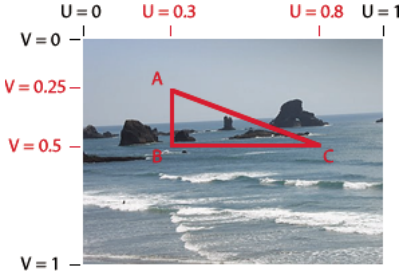
Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Depois que começar a trabalhar com texturas, você desejará usar o parâmetro uvtData de `drawTriangles()`. Esse parâmetro permite configurar o mapeamento UV para preenchimentos de bitmap.

O mapeamento UV é um método de aplicar textura a objetos. Ele se baseia em dois valores: um valor horizontal U (x) e um valor vertical V (y). Em vez de serem baseados em valores de pixel, eles são baseados em porcentagens. 0 U e 0 V é o canto superior esquerdo de uma imagem, e 1 U e 1 V é o canto inferior direito:

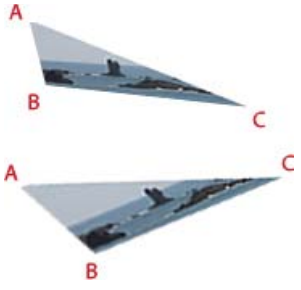
As localizações UV 0 e 1 em uma imagem de bitmap

Os vetores de um triângulo podem receber coordenadas UV para se associarem às respectivas localizações em uma imagem:



As coordenadas UV de uma área triangular de uma imagem de bitmap

Os valores UV permanecem consistentes com as pontas do triângulo:



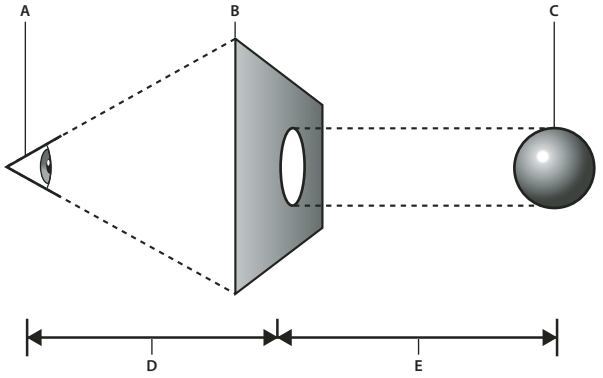
Os vértices do triângulo são movimentados e o bitmap é distorcido para manter iguais os valores UV de uma ponta individual

Conforme transformações 3D do ActionScript são aplicadas ao triângulo associado ao bitmap, a imagem do bitmap é aplicada ao triângulo com base nos valores UV. Portanto, em vez de usar cálculos de matriz, defina ou ajuste os valores UV para criar um efeito tridimensional.

O método `Graphics.drawTriangles()` também aceita uma informação opcional para transformações tridimensionais: o valor T. O valor T em uvtData representa a perspectiva 3D ou, mais especificamente, o fator de escala do vértice associado. O mapeamento UVT adiciona correção de perspectiva ao mapeamento UV. Por exemplo, se um objeto estiver posicionado no espaço 3D longe do ponto de visão, de modo que pareça ter 50% de seu tamanho “original”, o valor T desse objeto será 0,5. Uma vez que triângulos são desenhados para representar objetos no espaço 3D, sua localização no eixo z determina seus valores T. A equação que determina o valor T é a seguinte:

```
T = focalLength/(focalLength + z);
```

Nesta equação, focalLength representa uma distância focal ou uma localização "na tela" calculada, que determina o grau da perspectiva observado na exibição.



A distância focal e o valor z
***A.** ponto de visão **B.** tela **C.** objeto 3D **D.** valor de focalLength **E.** valor z*

O valor de T é usado para dimensionar formas básicas para fazer com que pareçam mais distantes. Normalmente é o valor usado para converter pontos 3D em pontos 2D. No caso de dados UVT, também é usado para dimensionar um bitmap entre as pontas de um triângulo com perspectiva.

Quando você define valores UVT, o valor T segue diretamente os valores UV definidos para um vértice. Com a inclusão de T, cada três valores no parâmetro `uvtData` (U, V e T) correspondem a cada dois valores no parâmetro `vertices` (x e y). Com valores UV apenas, uvtData.length == vertices.length. Com a inclusão de um valor T, uvtData.length = 1,5*vertices.length.

O exemplo a seguir mostra um plano sendo girado no espaço 3D usando dados UVT. Este exemplo usa uma imagem chamada ocean.jpg e uma classe “auxiliar” (ImageLoader) para carregar a imagem de maneira que ela possa ser atribuída ao objeto BitmapData.

Este é o código-fonte da classe ImageLoader (salve-o em um arquivo chamado 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);
    }
    }
}
```

E este é o ActionScript que usa triângulos, mapeamento UV e valores T para fazer com que a imagem pareça estar diminuindo à medida que se aproxima de um ponto de fuga e girando. Salve este código em um arquivo denominado 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
```

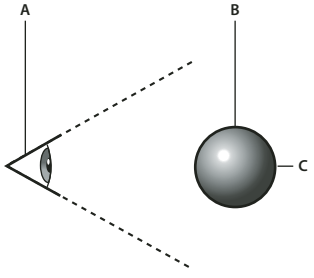
```
            t1 = focalLength/(focalLength + z1);
            t2 = focalLength/(focalLength + z2);
            t3 = focalLength/(focalLength + z3);
            t4 = focalLength/(focalLength + z4);

            // 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);
        }
    }
}
```

Para testar este exemplo, salve esses dois arquivos de classe no mesmo diretório de uma imagem chamada “ocean.jpg”. Você pode ver como o bitmap original é transformado para dar a impressão de que ele está desaparecendo conforme aumenta a distância e girando no espaço 3D.

Remoção

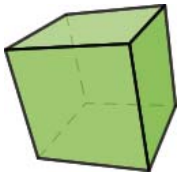
Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Remoção é o processo que determina quais superfícies de um objeto tridimensional o renderizador não deve renderizar pelo fato de estarem ocultas do ponto de visão atual. No espaço 3D, a superfície na “parte de trás” de um objeto tridimensional fica oculta do ponto de visão:



A parte de trás de um objeto 3D fica oculta do ponto de visão.
***A.** ponto de visão **B.** objeto 3D **C.** a parte de trás de um objeto tridimensional*

Inerentemente, todos os triângulos sempre são renderizados, seja qual for o tamanho, a forma ou a posição. A remoção assegura que o Flash Player ou o AIR renderize o objeto 3D corretamente. Além disso, para ganhar tempo nos ciclos de renderização, às vezes você deseja que alguns triângulos sejam ignorados pelo renderizador. Pense em um cubo girando no espaço. Em um determinado momento, você não verá mais do que três lados desse cubo, uma vez que os lados que não aparecem na exibição estariam voltados para a outra direção no outro lado do cubo. Como esses lados não serão vistos, o renderizados não deve desenhá-los. Sem a remoção, o Flash Player ou o AIR renderiza os lados da frente e de trás.



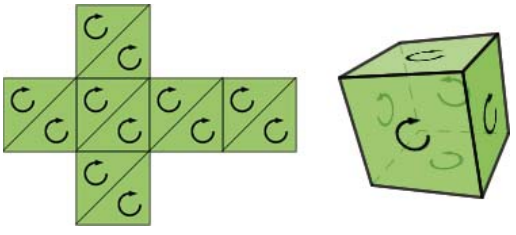
Um cubo tem lados que não ficam visíveis no ponto de visão atual

Por isso, o método `Graphics.drawTriangles()` tem um quarto parâmetro para estabelecer um valor de remoção:

```
public function drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null,
uvtData:Vector.<Number> = null, culling:String = "none"):void
```

O parâmetro de remoção é um valor da classe de enumeração `TriangleCulling`: `TriangleCulling.NONE`, `TriangleCulling.POSITIVE` e `TriangleCulling.NEGATIVE`. Esses valores dependem da direção do caminho do triângulo que define a superfície do objeto. A API do ActionScript para determinar a remoção pressupõe que todos os triângulos voltados para fora de uma forma 3D sejam desenhados com a mesma direção de caminho. Uma vez que uma face do triângulo é virada, a direção do caminho também muda. Nesse ponto, o triângulo pode ser removido (excluído da renderização).

Portanto, um valor `POSITIVE` de`TriangleCulling` remove triângulos com direção de caminho positiva (no sentido horário). Um valor `NEGATIVE` de `TriangleCulling` remove triângulos com uma direção de caminho negativa (no sentido anti-horário). No caso de um cubo, enquanto as superfícies voltadas para frente têm uma direção de caminho positiva, as superfícies traseiras têm uma direção de caminho negativa:



Um cubo "aberto" para mostrar a direção do caminho. Quando o cubo está "fechado", a direção do caminho da parte traseira é invertida.

Para ver como funciona a remoção, comece com o exemplo anterior de "Mapeamento UV" na página 374, defina o parâmetro de remoção do método `drawTriangles()` como `TriangleCulling.NEGATIVE`:

```
container.graphics.drawTriangles(vertices, indices, uvtData, TriangleCulling.NEGATIVE);
```

Observe que o lado "de trás" da imagem não é renderizado porque o objeto gira.

Capítulo 20: Princípios básicos do trabalho com texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para exibir texto na tela do Adobe® Flash® Player ou Adobe® AIR™, use uma instância da classe TextField ou use as classes Flash Text Engine. Essas classes permitem a criação, exibição e formatação de texto. Como alternativa, é possível usar a Text Layout Framework (TLF) - uma biblioteca de componentes que se baseia nas classes Flash Text Engine, porém projetada para simplificar sua utilização. Em dispositivos móveis, você pode usar a classe StageText para inserir texto.

É possível estabelecer conteúdo específico para campos de texto ou designar a origem do texto e definir sua aparência. Também é possível responder aos eventos do usuário assim que ele insira texto ou clique em um link de hipertexto.

A classe TextField e as classes Flash Text Engine lhe permitem exibir e gerenciar o texto no Flash Player e no AIR. Você pode usar a classe TextField para criar objetos de texto para exibição e inserção. TextField fornece os princípios para outros componentes baseados em texto, como TextArea e TextInput. É possível usar a classe TextFormat para definir a formatação dos caracteres e parágrafos dos objetos TextField e aplicar CSS (folhas de estilos em cascata) usando a propriedade Textfield.styeSheet e a classe StyleSheet. É possível designar texto formatado em HTML, que pode conter mídia incorporada (clipes de filme, arquivos SWF, arquivos GIF, arquivos PNG e arquivos JPEG), diretamente para um campo de texto.

O Flash Text Engine, disponível a partir do Flash Player 10 e do Adobe AIR 1.5, apresenta suporte em baixo nível para o controle sofisticado de métricas de texto, formatação e texto bidirecional. Oferece também fluxo de texto aprimorado e suporte avançado a idioma. Embora você possa usar o Flash Text Engine para criar e gerenciar os elementos de texto, ele foi projetado principalmente como base para criar componentes de tratamento de texto e exige maior conhecimento de programação. A Text Layout Framework, que inclui um componente de tratamento de texto baseado no Flash Text Engine, fornece uma forma facilitada de usar os recursos avançados do novo mecanismo de texto. A Text Layout Framework é uma biblioteca extensível totalmente integrada ao ActionScript 3.0. É possível usar o componente TLF existente ou usar a estrutura para construir seu próprio componente de texto.

A classe StageText, disponível a partir do AIR 3, oferece um campo nativo de entrada de texto. Como esse campo é fornecido pelo sistema operacional do dispositivo, ele oferece a experiência com a qual os usuários de um dispositivo estão mais familiarizados. Uma instância do StageText não é um objeto de exibição. Em vez de adicioná-la à lista de exibição, você atribui uma instância, um estágio e uma área de exibição ao estágio chamada de viewport. A instância do StageText é exibida na frente de qualquer objeto de exibição.

Para obter mais informações sobre esses tópicos, consulte:

- "Uso da classe TextField" na página 383
- "Uso do Flash Text Engine" na página 408
- "Uso da Text Layout Framework" na página 438
- Inserção de texto nativa com o StageText

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes envolvidos com o tratamento de texto:

Folhas de estilos em cascata Uma sintaxe padrão para especificação de estilos e formatação de conteúdo estruturado no formato XML (ou HTML).

Fonte do dispositivo Uma fonte instalada na máquina do usuário.

Campo de texto dinâmico Um campo de texto cujos conteúdos possam ser alterados pelo ActionScript, mas não pela entrada do usuário.

Fonte incorporada Uma fonte que tem seus dados de estrutura de tópicos de caracteres armazenados no arquivo SWF do aplicativo.

Texto HTML Conteúdo em texto inserido em um campo de texto usando o ActionScript que inclui tags de formatação HTML junto com o conteúdo de texto real.

Campo de texto de entrada Um campo de texto cujo conteúdo pode ser alterado pela entrada do usuário ou pelo ActionScript.

Kerning Um ajuste de espaço entre pares de caracteres a fim de tornar o espaçamento em palavras mais proporcional e o texto mais fácil de ler.

Campo de texto estático Um campo de texto criado na ferramenta de autoria, cujo conteúdo não pode ser alterado quando o arquivo SWF está em execução.

Métricas das linhas de texto Medições do tamanho de várias partes de texto em um campo de texto, por exemplo, a linha de base do texto, a altura do topo dos caracteres, o tamanho dos descendentes (a parte de algumas letras minúsculas que se estendem abaixo da linha de base) e assim por diante.

Tracking Um ajuste de espaço entre grupos de letras ou blocos de texto para aumentar ou diminuir a densidade e tornar o texto mais legível.

Capítulo 21: Uso da classe TextField

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar uma ocorrência da classe TextField para exibir texto ou criar um campo de entrada de texto na tela no Adobe® Flash® Player ou no Adobe® AIR™. A classe TextField é o ponto de partida para outros componentes com base em texto, como os componentes TextArea ou TextInput.

O conteúdo do campo de texto pode ser pré-especificado no arquivo SWF, carregado de um arquivo de texto, ou banco de dados, ou inserido por um usuário que interaja com o seu aplicativo. Em um campo de texto, o texto pode aparecer como conteúdo HTML renderizado, com imagens incorporadas em tal HTML. Depois de criar uma instância de um campo de texto, é possível usar as classes flash.text como TextFormat e StyleSheet, para controlar a aparência do texto. O pacote flash.text contém quase todas as classes relacionadas à criação, ao gerenciamento e à formatação de texto no ActionScript.

É possível formatar texto definindo a formatação com um objeto TextFormat e atribuir esse objeto no campo de texto. Se o campo de texto contiver texto HTML, será possível aplicar um objeto StyleSheet ao campo de texto para atribuir estilos a partes específicas do conteúdo do campo de texto. Os objetos TextFormat ou StyleSheet contêm propriedades que definem a aparência do texto, como cor, tamanho e peso. O objeto TextFormat designa as propriedades a todo o conteúdo dentro de um campo de texto para um intervalo de texto. Por exemplo, no mesmo campo, o texto de uma sentença pode estar em vermelho e negrito, e o da sentença seguinte em azul e itálico.

Além das classes no pacote flash.text, é possível usar a classe flash.events.TextEvent para responder às ações do usuário relacionadas ao texto.

Mais tópicos da Ajuda

"Atribuição de formatos de texto" na página 391

"Exibição de texto HTML" na página 385

"Aplicação de folhas de estilos em cascata" na página 391

Exibição de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Embora as ferramentas de autoria como o Adobe Flash Builder e o Flash Professional apresentem várias opções de exibição de texto, incluindo componentes relacionados a texto ou ferramentas de texto, o modo mais simples de exibir texto de modo programático é por meio de um campo de texto.

Tipos de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O tipo de texto dentro de um campo é caracterizado por sua fonte:

- Texto dinâmico

  O texto dinâmico inclui conteúdo carregado de uma fonte externa, por exemplo, um arquivo de texto, um arquivo XML ou mesmo um serviço da Web remoto.

- Texto de entrada

  Texto de entrada é qualquer texto inserido por um usuário ou texto dinâmico que um usuário possa editar. É possível configurar uma folha de estilos para formatar texto de entrada ou usar a classe flash.text.TextFormat para atribuir propriedades ao campo de texto do conteúdo de entrada. Para obter mais informações, consulte "Captura de entrada de texto" na página 389.

- Texto estático

  O texto estático é criado apenas através do Flash Professional. Não é possível criar uma instância de texto estático usando o ActionScript 3.0. No entanto, é possível usar classes ActionScript como StaticText e TextSnapshot para manipular uma instância de texto estática. Para obter mais informações, consulte "Trabalho com texto estático" na página 397

Modificação de conteúdos de campo de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode definir texto dinâmico atribuindo uma sequência de caracteres à propriedade `flash.text.TextField.text`. Você designa uma sequência de caracteres diretamente à propriedade, como segue:

```
myTextField.text = "Hello World";
```

Também é possível designar à propriedade `text` um valor de uma variável definida no script, como no exemplo a seguir:

```
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;
        }
    }
}
```

Como alternativa, você pode designar à propriedade `text` um valor de uma variável remota. Você tem três opções para carregar valores de texto de fontes remotas:

- As classes flash.net.URLLoader e flash.net.URLRequest carregam as variáveis do texto da posição local para uma remota.
- O atributo `FlashVars` está incorporado na página HTML que hospeda o arquivo SWF e pode conter valores das variáveis de texto.
- A classe flash.net.SharedObject gerencia armazenamento persistente de valores. Para obter mais informações, consulte "Armazenamento de dados locais" na página 724.

Exibição de texto HTML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe flash.text.TextField tem uma propriedade `htmlText` que pode ser usada para identificar a sequência de texto à medida que ela contém tags HTML para formatação de conteúdo. Como no exemplo a seguir, atribua um valor de sequência de caracteres à propriedade `htmlText` (não à propriedade `text`) do Flash Player ou AIR para renderizar o texto como HTML:

```
var myText:String = "<p>This is <b>some</b> content to <i>render</i> as <u>HTML</u> text.</p>";
myTextBox.htmlText = myText;
```

O Flash Player e o AIR dão suporte a um subconjunto de tags HTML e às entidades da propriedade `htmlText`. A descrição da propriedade `flash.text.TextField.htmlText` na Referência do ActionScript 3.0 apresenta informações detalhadas sobre as tags e entidades HTML com suporte.

Assim que projetar o conteúdo usando a propriedade `htmlText`, você poderá usar as folhas de estilos ou a tag `textformat` para gerenciar a formatação do conteúdo. Para obter mais informações, consulte "Formatação de texto" na página 391.

Uso de imagens em campos de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Outra vantagem de exibir o conteúdo como texto HTML é a inclusão de imagens no campo de texto. É possível fazer referência a uma imagem, local ou remota, usando a tag `img` e fazer com que ela apareça dentro do campo de texto associado.

O exemplo a seguir cria um campo de texto denominado `myTextBox` e inclui a imagem JPG de um olho, armazenada no mesmo diretório que o arquivo SWF, dentro do texto exibido:

```
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;
        }
    }
}
```

A tag `img` suporta arquivos JPEG, GIF, PNG e SWF.

Rolagem de texto em um campo de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível que, em muitos casos, o texto seja maior que o campo de exibição. Ou talvez um campo de entrada permita que um usuário insira mais texto do que possa ser exibido de uma única vez. Nesses casos, use as propriedades da classe flash.text.TextField relativas à rolagem para gerenciar conteúdos extensos, quer na vertical quanto na horizontal.

Essas propriedades incluem `TextField.scrollV`, `TextField.scrollH`, `maxScrollV` e `maxScrollH`. Use essas propriedades para responder a eventos, como um clique de mouse ou um pressionamento de tecla.

O exemplo a seguir cria um campo de texto que é um tamanho definido e contém mais texto do que o campo pode exibir de uma única vez. O texto rola verticalmente à medida que o usuário clica no campo de texto.

```
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++;
        }
    }
}
```

Seleção e manipulação de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível selecionar texto dinâmico ou de entrada. Como as propriedades e métodos de seleção de texto da classe TextField usam posições de indexação para definir o intervalo do texto a ser manipulado, é possível selecionar programaticamente o texto dinâmico ou de entrada mesmo quando o conteúdo é desconhecido.

Nota: *No Flash Professional, se você escolher a opção selecionável em um campo de texto estático, o campo de texto a ser exportado e colocado na lista de exibição será um campo de texto regular, dinâmico.*

Seleção de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A propriedade `flash.text.TextField.selectable` é `true` por padrão e é possível selecionar texto programaticamente usando o método `setSelection()`.

Por exemplo, é possível definir um texto específico dentro de um campo de texto para ser selecionado quando o usuário clicar no campo de texto:

```
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);
}
```

De modo similar, para que o texto dentro de um campo de texto seja selecionado assim que o texto for inicialmente exibido, crie uma função do manipulador de eventos que seja chamada quando o campo de texto for adicionado à lista de exibição.

Captura de texto selecionado pelo usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As propriedades `selectionBeginIndex` e `selectionEndIndex` de TextField (que são "somente leitura" e, portanto, não podem ser definidas para selecionar texto de modo programático) podem ser usadas para capturar qualquer coisa que o usuário tenha selecionado atualmente. Adicionalmente, os campos de texto de entrada podem usar a propriedade `caretIndex`.

Por exemplo, o código a seguir controla os valores dos índices do texto selecionado pelo usuário:

```
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);
}
```

É possível aplicar uma coleção das propriedades do objeto TextFormat para a seleção a fim de alterar a aparência do texto. Para obter mais informações sobre a aplicação de uma coleção de propriedades TextFormat para o texto selecionado, consulte "Formatação de intervalos de texto dentro de um campo de texto" na página 394.

Captura de entrada de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Por padrão, a propriedade `type` de um campo de texto está definida como `dynamic`. Se você definir a propriedade `type` como `input` usando a classe TextFieldType, poderá colecionar a entrada do usuário e salvar o valor para uso em outras partes do aplicativo. Os campos de texto de entrada são úteis para formulários e quaisquer aplicativos que desejem que o usuário defina um valor de texto a ser utilizado em outro lugar no programa.

Por exemplo, o código a seguir cria um campo de texto de entrada denominado `myTextBox`. Assim que o usuário inserir texto no campo, o evento `textInput` será acionado. Um manipulador de eventos denominado `textInputCapture` capturará a sequência de caracteres de texto inserida e a designará para uma variável. O Flash Player ou AIR exibirá o novo texto em outro campo de texto, denominado `myOutputBox`.

```
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;
        }

    }
}
```

Restrição de entrada de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Como campos de texto de entrada são usados frequentemente para formulários ou caixas de diálogos nos aplicativos, talvez você queira limitar os tipos de caracteres que um usuário pode inserir em um campo de texto ou mantê-lo oculto - por exemplo, no caso de uma senha. A classe flash.text.TextField tem as propriedades `displayAsPassword` e `restrict` que podem ser definidas para controlar a entrada do usuário.

A propriedade `displayAsPassword` simplesmente oculta o texto (exibindo-o como uma série de asteriscos) à medida que o usuário o digita. Quando `displayAsPassword` está definido como `true`, os comandos Recortar e Copiar, e seus respectivos atalhos no teclado, não funcionam. Como mostrado no exemplo a seguir, designe a propriedade `displayAsPassword` exatamente como faria com outras propriedades, como background e color:

```
myTextBox.type = TextFieldType.INPUT;
myTextBox.background = true;
myTextBox.displayAsPassword = true;
addChild(myTextBox);
```

A propriedade `restrict` é um pouco mais complicada visto que você precisa especificar que caracteres o usuário tem permissão para digitar em um campo de texto de entrada. As entradas admissíveis são letras e números específicos ou intervalos de letras, números e caracteres. O código a seguir permite que o usuário insira letras maiúsculas apenas (e não números ou caracteres especiais) no campo de texto:

```
myTextBox.restrict = "A-Z";
```

O ActionScript 3.0 usa hífens para definir intervalos e circunflexos para definir caracteres excluídos. Para obter mais informações sobre a como definir o que está restrito em um campo de texto de entrada, consulte a entrada da propriedade `flash.text.TextField.restrict` na Referência do ActionScript 3.0.

Nota: *Se você utilizar a propriedade `flash.text.TextField.restrict`, o runtime converterá automaticamente letras restritas no caso permitido. Se você usar a propriedade `fl.text.TLFTextField.restrict` (isto é, se você usar um campo de texto TLF), o runtime ignorará as letras restritas.*

Formatação de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você tem várias opções para formatar programaticamente a exibição do texto. É possível definir propriedades diretamente na instância do TextField — por exemplo, as propriedades `TextFIeld.thickness`, `TextField.textColor` e `TextField.textHeight`.Ou você pode designar o conteúdo do campo de texto usando a propriedade `htmlText` e lançar mão das tags HTML com suporte, por exemplo `b`, `i` e `u`. No entanto, também é possível aplicar objetos TextFormat para os campos de texto que contém texto sem formatação ou objetos StyleSheet para os campos de texto que contêm a propriedade `htmlText`. O uso dos objetos TextFormat e StyleSheet fornece a maioria do controle e consistência sobre a aparência do texto em todo o aplicativo. É possível definir um objeto TextFormat ou StyleSheet e aplicá-lo a vários ou a todos os campos de texto no seu aplicativo.

Atribuição de formatos de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar a classe TextFormat para definir um número diferente de propriedade de exibição de texto e aplicá-las à todo o conteúdo de um objeto TextField ou a um intervalo de texto.

O exemplo a seguir aplica um objeto de TextFormat a um objeto TextField inteiro e aplica um segundo Objeto TextFormat a um intervalo de texto dentro desse objeto TextField.

```
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);
```

O método `TextField.setTextFormat()` afeta apenas o texto que já está exibido no campo de texto. Se o conteúdo em TextField mudar, talvez seja necessário que o aplicativo chame o método `TextField.setTextFormat()` novamente para reaplicar a formatação. Ainda é possível definir a propriedade `defaultTextFormat` de TextField para especificar o formato a ser usado para o texto inserido pelo usuário.

Aplicação de folhas de estilos em cascata

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os campos de texto contêm texto sem formatação ou formatado em HTML. O texto sem formatação está armazenado na propriedade `text` da instância e o texto HTML está armazenado na propriedade`htmlText`.

É possível usar declarações de estilo CSS para definir estilos de texto que você possa aplicar a diferentes campos de texto. As declarações de estilo CSS podem ser criadas no seu código de aplicativo ou carregada em um tempo de execução a partir de um arquivo CSS externo.

A classe flash.text.StyleSheet manipula os estilos das CSS. A classe StyleSheet reconhece um conjunto limitado de propriedades das CSS. Para obter uma lista detalhada das propriedades de estilo que a classe StyleSheet suporta, consulte a entrada flash.textStylesheet na Referência do ActionScript 3.0.

Como o exemplo a seguir mostra, é possível criar as CSS em seu código e aplicar esses estilos ao texto HTML usando um objeto StyleSheet:

```
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);
```

Depois de criar um objeto StyleSheet, o código de exemplo cria um objeto simples para manter um conjunto de propriedades de declaração de estilos. Em seguida, ele chama o método `StyleSheet.setStyle()` que adiciona o novo estilo à folha de estilos com o nome ".darkred". Depois, ele aplica a formatação da folha de estilos atribuindo o objeto StyleSheet à propriedade `styleSheet` do TextField.

Para que os estilos das CSS tenham efeito, a folha de estilos deve ser aplicada ao objeto TextField antes que a propriedade `htmlText` seja definida.

Por design, um campo de texto com uma folha de estilos não é editável. Se tiver um campo de texto de entrada e atribuir uma folha de estilos a ele, o campo de texto apresentará as propriedades da folha de estilos, mas o campo texto não permitirá que usuários insiram o novo texto nele. Além disso, não será possível usar as APIs do ActionScript abaixo em um campo de texto caso uma folha de estilos esteja atribuída:

- O método `TextField.replaceText()`
- O método `TextField.replaceSelectedText()`
- A propriedade `TextField.defaultTextFormat`
- O método `TextField.setTextFormat()`

Se um campo de texto tiver uma folha de estilos atribuída a ele, no entanto, posteriormente, a propriedade `TextField.styleSheet` for definida como `null`, o conteúdo das propriedades `TextField.text` e `TextField.htmlText` adicionarão tags e atributos ao conteúdo para incorporar a formatação da folha de estilos anteriormente atribuída. A fim de preservar a propriedade `htmlText` original, salve-a em uma variável antes de definir a folha de estilos como `null`.

Carregamento de um arquivo CSS externo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A abordagem das CSS para formatação será mais eficaz quando for possível carregar as informações das CSS desde um arquivo externo no tempo de execução. Quando os dados das CSS forem externos ao próprio aplicativo, será possível alterar o estilo visual do texto no aplicativo sem ter de alterar o código de origem do ActionScript 3.0. Depois da implementação do aplicativo, é possível alterar um arquivo CSS externo para alterar a aparência do aplicativo, sem ter de reimplementar o arquivo SWF do aplicativo.

O método `StyleSheet.parseCSS()` converte uma sequência de caracteres que contém dados CSS em declarações de estilo no objeto StyleSheet. O exemplo a seguir mostra como ler um arquivo CSS externo e aplicar suas declarações de estilo a um objeto TextField.

Em primeiro lugar, este é o conteúdo do arquivo CSS a ser carregado, denominado exemplo.css:

```
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;
}
```

Ao lado, está o código do ActionScript para uma classe que carrega o arquivo example.css e aplica os estilos ao conteúdo TextField.

```
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;
        }
    }
}
```

Quando os dados das CSS forem carregados, o método `onCSSFileLoaded()` será executado e chamará o método `StyleSheet.parseCSS()` para transferir as declarações de estilos para o objeto StyleSheet.

Formatação de intervalos de texto dentro de um campo de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um método útil da classe flash.text.TextField é o método `setTextFormat()` Com `setTextFormat()`, é possível atribuir propriedades específicas ao conteúdo de parte de um campo de texto para responder à entrada do usuário, por exemplo, formulários que precisem lembrar usuários que determinadas entradas são necessárias ou alterar a ênfase de uma subseção de uma passagem de texto em um campo quando um usuário seleciona partes do texto.

O exemplo a seguir usa `TextField.setTextFormat()` em um cadeia de caracteres para alterar a aparência de parte do conteúdo do `myTextField` quando o usuário clica no campo de texto:

```
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);
}
```

Renderização avançada de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 apresenta uma variedade de classes no pacote flash.text para controlar as propriedades do texto exibido, incluindo fontes incorporadas, configurações de suavização de serrilhado, controle de canal alfa e outras configurações específicas. A Referência do ActionScript 3.0 apresenta descrições detalhadas das classes e propriedades, incluindo as classes CSMSettings, Font e TextRenderer.

Uso de fontes incorporadas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando você especificar uma fonte específica para um TextField no aplicativo, o Flash Player ou AIR procurará uma fonte de dispositivo (uma fonte que reside no computador do usuário) com o mesmo nome. Se essa fonte não for localizada no sistema ou se o usuário tiver uma versão ligeiramente diferente de uma fonte com esse nome, a aparência da exibição do texto poderá ser bem diferente do que se pretende. Por padrão, o texto aparece em uma fonte Times Roman.

Para ter certeza de que o usuário verá exatamente a fonte correta, incorpore a fonte no arquivo SWF do aplicativo. As fontes incorporadas têm um número de benefícios:

- Os caracteres de fonte incorporada têm suavização de serrilhado, o que faz com que as bordas sejam exibidas com mais suavidade, especialmente em um texto maior.
- É possível girar texto que use fontes incorporadas.
- Texto com fonte incorporada pode ser transparente ou semitransparente.
- É possível usar o estilo CSS `kerning` com fontes incorporadas.

A maior limitação ao uso de fontes incorporadas é que elas aumentam o tamanho do arquivo ou do download do aplicativo.

O método exato a ser usado para incorporar um arquivo fonte no arquivo SWF do aplicativo varia de acordo com o ambiente de desenvolvimento.

Depois de incorporar uma fonte, é possível verificar se um TextField utilizará a fonte incorporada correta:

- Defina a propriedade `embedFonts` do TextField como `true`.
- Crie um objeto TextFormat, defina a propriedade `fontFamily` como o nome da fonte incorporada e aplique o objeto TextFormat ao TextField. Quando você especificar uma fonte incorporada, a propriedade `fontFamily` deverá conter apenas um único nome, e não poderá usar uma lista delimitada por vírgulas de vários nomes de fonte.
- Se estiver usando os estilos CSS para definir fontes para os TextFields ou componentes, defina a propriedade das CSS `font-family` para o nome da fonte incorporada. A propriedade `font-family` deverá conter um único nome, não uma lista de nomes, se você desejar especificar uma fonte incorporada.

Incorporação de uma fonte no Flash

O Flash Professional permite que você incorpore qualquer fonte que possua ao sistema, incluindo as fontes True Type e Postscript Tipo 1.

É possível incorporar fontes em um aplicativo de muitos modos, incluindo:

- Configurando as propriedades da fonte e do estilo de um TextField no Palco e clicando na caixa de seleção Fontes incorporadas

- Criando e fazendo referência a um símbolo de fonte
- Criando e usando uma biblioteca compartilhada de tempo de execução que contém símbolos de fonte incorporada

Para obter detalhes adicionais sobre como incorporar fontes em aplicativos, consulte "Fontes incorporadas para campos de texto dinâmicos ou de entrada" em *Uso do Flash*.

Incorporação de uma fonte no Flex

É possível incorporar fontes em um aplicativo Flex de muitos modos, incluindo:

- Usar a tag de metadados `[Embed]` em um script
- Usar a declaração de estilo `@font-face`
- Estabeleça uma classe para a fonte e utilize a tag `[Embed]` para integrá-la.

Apenas fontes True Type podem ser diretamente incorporadas em um aplicativo Flex. Para fontes em outros formatos (como Postscript Tipo 1), é possível incorporá-las, primeiro, em um arquivo SWF e, depois, usando o Flash Professional, usar tal arquivo no seu aplicativo Flex. Para obter detalhes sobre como usar fontes incorporadas dos arquivos SWF no Flex, consulte "Incorporação de fontes de arquivos SWF" em *Uso do Flex 4*.

Mais tópicos da Ajuda

Incorporar fontes para proporcionar uma aparência uniforme ao texto

Peter deHaan: Embedding fonts (Incorporação de Fontes)

Divillysausages.com: AS3 Font embedding masterclass (Masterclass de Incorporação de Fontes do AS3)

Controle de nitidez, espessura e suavização de serrilhado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Por padrão, o Flash Player ou AIR determina as configurações dos controles de exibição de texto como nitidez, espessura e suavização de serrilhado à medida que o texto é redimensionado, muda de cor ou exibido em diversos fundos. Em alguns casos (por exemplo, quando se tem um texto muito pequeno, muito grande ou uma diversidade de fundos exclusivos), talvez você queira manter controle sobre essas configurações. É possível substituir as configurações do Flash Player ou AIR usando a classe `flash.text.TextRenderer` e suas classes associadas, como a classe CSMSettings. Essas classes concedem controle preciso sobre a renderização da qualidade do texto incorporado. Para obter informações adicionais sobre fontes incorporadas, consulte “Uso de fontes incorporadas” na página 395

Nota: *A propriedade `flash.text.TextField`.antiAliasType deve ter o valor `AntiAliasType.ADVANCED` para configurar a nitidez, espessura ou propriedade gridFitType, ou usar o método `TextRenderer.setAdvancedAntiAliasingTable()`.*

O exemplo a seguir aplica-se às propriedades e formatação da CSM (modulação de traçado contínuo) personalizadas para o texto exibido usando uma fonte incorporada denominada `myFont`. Quando o usuário clicar no texto exibido, o Flash Player ou o Adobe AIR aplicará as configurações personalizadas.

```
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);
}
```

Trabalho com texto estático

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O texto estático só é criado no Flash Professional. Não é possível instanciar programaticamente texto estático usando ActionScript. O texto estático é útil quando o texto é pequeno e não pode ser alterado (como o texto dinâmico pode). Pense no texto estático como a um elemento gráfico (por exemplo, um círculo ou quadrado) desenhado no Palco, no Flash Professional. Embora o texto estático seja mais limitado que o texto dinâmico, o ActionScript 3.0 permite que você leia os valores da propriedade do texto estático usando a classe StaticText. É possível também usar a classe TextSnapshot para ler outros tipos de valores além do texto estático.

Acesso aos campos de texto estático com a classe StaticText

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Normalmente, é possível usar a classe flash.text.StaticText no painel Ações do Flash Professional para interagir com uma instância de texto estática posicionada no Palco. Também é possível trabalhar nos arquivos do ActionScript que interagem com um arquivo SWF que contém texto estático. Em qualquer caso, é possível instanciar uma instância de texto estático de modo programático. O texto estático é criado no Flash Professional.

Para criar uma referência a um campo de texto estático existente, faça uma iteração nos itens na lista de exibição e atribua um variável. Por exemplo:

```
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);
}
}
```

Depois de fazer referência a um campo de texto estático, é possível usar as propriedades desse campo no ActionScript 3.0. O código a seguir está anexado a um quadro na Linha de tempo e parte da premissa de que uma variável denominada `myFieldLabel` foi atribuída a uma referência a texto estático. Um campo de texto dinâmico, denominado `myField`, está posicionado em relação aos valores `x` e `y` de `myFieldLabel` e exibe o valor de `myFieldLabel` novamente.

```
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
```

Uso da classe TextSnapshot

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se desejar trabalhar programaticamente com uma instância de texto estática existente, poderá usar a classe flash.text.TextSnapshot para trabalhar com a propriedade `textSnapshot` de um flash.display.DisplayObjectContainer. Em outras palavras, você cria uma instância do TextSnapshot da propriedade `DisplayObjectContainer.textSnapshot` É possível aplicar métodos a essa instância para recuperar valores ou selecionar partes do texto estático.

Por exemplo, coloque um campo de texto estático que contém o texto "Exemplo de TextSnapshot" no Palco. Adicione o ActionScript a seguir ao Quadro 1 da Linha de tempo:

```
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);
```

A classe TextSnapshot será útil para obter texto dentre os campos de texto estático em um arquivo SWF carregado, caso você queira usar o texto como um valor em outra parte de um aplicativo.

Exemplo de TextField: formatação de texto estilo jornal

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo Layout de notícias formata o texto para uma aparência similar a uma matéria de jornal impresso. O texto de entrada pode conter um título, um subtítulo e o corpo da matéria. Com determinada largura e altura de exibição, o exemplo de Layout de notícias formata o título e o subtítulo de modo a ocupar toda a extensão da área de exibição. O texto da matéria é distribuído em duas ou mais colunas.

Este exemplo ilustra as seguintes técnicas de programação do ActionScript:

- Extensão da classe TextField
- Carregamento e aplicação de um arquivo CSS externo
- Conversão de estilos CSS em objetos TextFormat
- Uso da classe TextLineMetrics para obter informações sobre o tamanho de exibição de texto

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Layout de notícias podem ser encontrados na pasta Samples/NewsLayout. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
NewsLayout.mxml ou NewsLayout.fla	A interface do usuário do aplicativo para Flex (MXML) ou Flash (FLA).
com/example/programmingas3/newslayout/StoryLayoutComponent.as	Uma classe UIComponent do Flex que substitui a instância do StoryLayout.
com/example/programmingas3/newslayout/StoryLayout.as	A principal classe do ActionScript que organiza todos os componentes de uma matéria jornalística para exibição.
com/example/programmingas3/newslayout/FormattedTextField.as	Uma subclasse da classe TextField que gerencia seus próprio TextFormat.
com/example/programmingas3/newslayout/HeadlineTextField.as	Uma subclasse da classe FormattedTextField que ajusta os tamanhos das fontes de modo a se ajustarem à largura desejada.
com/example/programmingas3/newslayout/MultiColumnTextField.as	Uma classe ActionScript que divide texto em duas ou mais colunas.
story.css	Um arquivo CSS que define estilos de texto para o layout.

Leitura do arquivo CSS externo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo Layout de notícias inicia pela leitura do texto da matéria de um arquivo XML local. Em seguida, ele lê um arquivo CSS externo que fornece as informações de formatação para o título, subtítulo e texto principal.

O arquivo CSS define três estilos, um estilo de parágrafo padrão para a matéria, e os estilos h1 e h2 para título e subtítulo respectivamente.

```
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;
}
```

A técnica usada par ler o arquivo CSS externo é a mesma descrita em "Carregamento de um arquivo CSS externo" na página 392 Quando o arquivo CSS for carregado, o aplicativo executará o método `onCSSFileLoaded()`, como mostrado abaixo.

```
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();
}
```

O método `onCSSFileLoaded()` cria um objeto StyleSheet e o utiliza para analisar os dados das CSS de entrada. O principal texto da matéria é exibida em um objeto MultiColumnTextField, que pode usar um objeto StyleSheet diretamente. No entanto, os campos do título usam a classe HeadlineTextField, que usam um objeto TextFormat para formatação.

O método `onCSSFileLoaded()` chama o método `getTextStyle()` duas vezes para converter uma declaração de estilo CSS em um objeto TextFormat a ser utilizado com cada um dos dois objetos TextField do título.

```
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;
}
```

Os nomes das propriedades e o significado dos valores das propriedades são diferentes entre as declarações de estilo CSS e os objetos TextFormat. O método `getTextStyle()` converte os valores das propriedades das CSS nos valores esperados pelo objeto TextFormat.

Organização dos elementos da matéria na página

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe StoryLayout formata e dispõe o título, o subtítulo e os campos de texto principais em uma organização com estilo de jornal. O método `displayText()` cria inicialmente e posiciona os diversos campos.

```
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;
...
```

Cada campo é posicionado abaixo do campo anterior definindo sua propriedade `y` de modo a igualar a propriedade `y` do campo anterior mais sua altura. Esse cálculo dinâmico de posicionamento é necessário porque os objetos HeadlineTextField e MultiColumnTextField podem alterar sua altura de modo a se adequarem ao conteúdo.

Alteração do tamanho da fonte para ajustar o tamanho do campo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Dada a largura em pixels e o número máximo de linhas, o HeadlineTextField altera o tamanho da fonte de modo a ajustar o texto ao campo. Se o texto for curto, o tamanho da fonte será grande, criando um título estilo tabloide. Se o texto for longo, o tamanho da fonte será menor.

O método `HeadlineTextField.fitText()` mostrado abaixo faz com que o dimensionamento de fonte funcione:

```
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;
    }
}
```

O método `HeadlineTextField.fitText()` usa uma técnica recursiva simples para dimensionar a fonte. Em primeiro lugar, ele advinha um número médio de pixels por caractere no texto e, então, calcula um tamanho de ponto inicial. Em seguida, altera o tamanho da fonte e verifica se o texto tem quebra de palavras para criar mais do que o número máximo de linhas de texto. Se houver linhas em excesso, ele chamará o método `shrinkText()` para diminuir o tamanho da fonte e tentar novamente. Se não houver linhas em excesso, ele chamará o método `growText()` para aumentar o tamanho da fonte e tentar novamente. O processo parará no ponto em que incrementar o tamanho da fonte em mais um criará linhas em excesso.

Divisão de texto em várias colunas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe MultiColumnTextField expande o texto entre vários objetos TextField que são organizados, por sua vez, como colunas de jornal.

O construtor `MultiColumnTextField()` cria, primeiro, uma matriz de objetos TextFields, uma para cada coluna, conforme mostrado aqui:

```
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);
}
```

Cada objeto TextField é adicionado à matriz e à lista de exibição com o método `addChild()`.

Sempre que a propriedade `text` ou `styleSheet` do StoryLayout muda, ele chama o método `layoutColumns()` para re-exibir o texto. O método `layoutColumns()` chama o método `getOptimalHeight()` para calcular a altura em pixels correta necessária para ajustar todo o texto a determinada largura de layout.

```
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;
    }
}
```

Em primeiro lugar, o método `getOptimalHeight()` calcula a largura de cada coluna. Em seguida, define a largura e a propriedade `htmlText` do primeiro objeto TextField na matriz. O método `getOptimalHeight()` usa esse primeiro objeto TextField para descobrir o número total de linhas com quebra de palavras no texto e, com base nesse número, identifica quantas linhas deve haver em cada coluna. Em seguida, ele chama o método `TextField.getLineMetrics()` para recuperar um objeto TextLineMetrics contendo detalhes sobre o tamanho do texto na primeira linha. A propriedade `TextLineMetrics.height` representa a altura completa de uma linha de texto, em pixels, incluindo ascendente, descendente e entrelinha. A altura ideal do objeto MultiColumnTextField é a altura da linha multiplicada pelo número de linhas por coluna, mais 4 para computar a borda de dois pixels em cima e embaixo de um objeto TextField.

Este é o código do método `layoutColumns()` completo:

```
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);
        }
    }
    }
}
```

Depois da propriedade `preferredHeight` ter sido definida pela chamada do método `getOptimalHeight()`, o método `layoutColumns()` será iterado por meio dos objetos TextField definindo a altura de cada objeto para o valor `preferredHeight`. O método `layoutColumns()` distribui apenas linhas suficientes de texto para cada campo de modo que não ocorra rolamento em nenhum campo individual e o texto em cada campo sucessivo inicie onde termina o texto no campo anterior. Se o estilo de alinhamento de texto for definido como “justificar”, o método `justifyLastLine()` será chamado para justificar a linha final do texto em um campo. Caso contrário, a última linha será tratada como uma linha de fim de parágrafo e não justificada.

Capítulo 22: Uso do Flash Text Engine

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O Adobe® Flash® Text Engine (FTE), disponível a partir do Flash Player 10 e do Adobe® AIR™1.5, fornece um suporte de baixo nível para o controle sofisticado de métricas de texto, formatação e texto bidirecional. Oferece fluxo de texto aprimorado e suporte avançado a idioma. Embora possa ser usado para criar e gerenciar elementos de texto simples, o FTE foi desenvolvido principalmente como uma base para os desenvolvedores criarem componentes de tratamento de texto. Assim, o Flash Text Engine presume um nível mais avançado de especialização em programação. Para exibir elementos de texto simples, consulte "Uso da classe TextField" na página 383.

A Text Layout Framework, que inclui um componente de tratamento de texto baseado na FTE, fornece uma forma mais fácil de usar seus recursos avançados. A Text Layout Framework é uma biblioteca extensível totalmente integrada ao ActionScript 3.0. É possível usar o componente TLF existente ou usar a estrutura para construir seu próprio componente de texto. Para obter mais informações, consulte "Uso da Text Layout Framework" na página 438.

Mais tópicos da Ajuda

pacote flash.text.engine

Criação e exibição de texto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Essas classes que compõe o Flash Text Engine permitem a criação, formatação e controle de texto. As classes a seguir são blocos de construção básicos para criação e exibição de texto com o Flash Text Engine:

- TextElement/GraphicElement/GroupElement - contém o conteúdo de uma instância do TextBlock
- ElementFormat - especifica os atributos de formatação do conteúdo de uma instância do TextBlock
- TextBlock - a fábrica para construção de um parágrafo de texto
- TextLine - uma linha de texto criada a partir do TextBlock

Para exibir texto, crie um objeto TextElement de um String usando um objeto ElementFormat para especificar as características de formatação. Atribua o TextElement à propriedade `content` de um objeto TextBlock. Crie as linhas de texto da exibição chamando o método `TextBlock.createTextLine()`. O método `createTextLine()` retorna um objeto TextLine que contém tantas strings quantas vão se ajustar na largura especificada. Chame o método repetidamente até que toda a string tenha sido formatada em linhas. Quando não houver mais linhas para criar, à propriedade textLineCreationResult do objeto TextBlock é atribuído o valor: `TextLineCreationResult.COMPLETE`. Para mostrar as linhas, acrescente-as à lista de exibição (com os valores apropriados da posição `x` e `y`).

O código a seguir, por exemplo, usa as classes FTE para exibir "Hello World!". This is Flash Text Engine!" (Olá a todos, esse é o Flash Text Engine), usando o formato e os valores de fonte padrão. Neste exemplo simples, é criada apenas uma linha de texto.

```
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;
        }
    }
}
```

Os parâmetros para `createTextLine()` especificam a linha a partir de qual inicia a nova linha e a largura da linha em pixels. A linha a partir da qual se inicia a nova linha é, normalmente, a linha anterior, contudo, no caso da primeira linha, ela é `null`.

Adição dos objetos GraphicElement e GroupElement

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível atribuir um objeto GraphicElement para um objeto TextBlock a fim de exibir uma imagem ou um elemento gráfico. Simplesmente, crie uma instância da classe GraphicElement de um gráfico ou de uma imagem e atribua a instância para a propriedade `TextBlock.content`. Crie a linha de texto chamando `TextBlock.createTextline()` como você faz normalmente. O exemplo a seguir cria duas linhas de texto, uma com um objeto GraphicElement e outra com um objeto TextElement.

```
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;
        }
    }
}
```

Você pode criar um objeto GroupElement para criar um grupo de objetos TextElement, GraphicElement, e outros objetos GroupElement. Um GroupElement pode ser atribuído à propriedade `content` de um objeto TextBlock. O parâmetro para o construtor `GroupElement()` é um Vetor, que aponta para os elementos de texto, gráficos e de grupo que formam o grupo. O exemplo a seguir agrupa dois elementos gráficos e um elemento de texto e os designa como unidade para um bloco de texto.

```
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;
        }
    }
}
```

Substituição de texto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível substituir texto em uma ocorrência de TextBlock chamando `TextElement.replaceText()` para substituir o texto no TextElement que você atribuiu à propriedade `TextBlock.content`.

O exemplo a seguir usa `replaceText()` para primeiro inserir texto no início da linha, depois anexar texto ao final da linha e por fim substituir texto no meio da linha.

```
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);
        }
    }
}
```

O método `replaceText()` substitui o texto especificado pelos parâmetros `beginIndex` e `endIndex` pelo texto especificado pelo parâmetro `newText`. Se os valores dos parâmetros `beginIndex` e `endIndex` forem idênticos, `replaceText()` inserirá o texto especificado nesse local. Caso contrário, ele substituirá pelo novo texto os caracteres especificados por `beginIndex` e `endIndex`.

Manipulação de eventos no FTE

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível adicionar ouvintes de eventos a uma instância do TextLine assim como a outros objetos de exibição. Por exemplo, é possível detectar quando um usuário passa o mouse sobre uma linha de texto ou um usuário clica na linha. O exemplo a seguir detecta ambos os eventos. Quando o usuário passar o mouse sobre a linha, o cursor mudará para um cursor de botão, e quando ele clicar na linha, a cor do cursor mudará.

```
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);
        }
    }
}
```

Espelhamento de eventos

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Também é possível espelhar os eventos em um bloco de texto ou em uma parte de um bloco de texto para um dispatcher de eventos. Em primeiro lugar, crie uma instância do EventDispatcher e a atribua à propriedade `eventMirror` de uma instância do TextElement. Se o bloco for composto de um único elemento de texto, o mecanismo espelhará eventos para todo o bloco de texto. Se o texto for composto de vários elementos de texto, o mecanismo espelhará eventos apenas para as instâncias do TextElement que tenham a propriedade `eventMirror` definida. O texto no exemplo a seguir é composto por três elementos: a palavra "Click" (Clique), a palavra "here" (aqui) e a sequência de caracteres "to see me in italic" (para me ver em itálico). O exemplo designa um dispatcher de eventos ao segundo elemento de texto, a palavra "here" e adiciona um ouvinte de eventos, o método `clickHandler()`. O método `clickHandler()` altera o texto para itálico. Ele substitui também o conteúdo do terceiro elemento para rezar "Click here to see me in normal font!" (Clique aqui para me ver em fonte normal!)

```
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);
        }
    }
}
```

As funções `mouseOverHandler()` e `mouseOutHandler()` muda o cursor para um cursor de botão quando ele está passando sobre a palavra "here" e o retorna para uma seta quando não está.

Formatação de texto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Um objeto `TextBlock` é uma fábrica para criação de linhas de texto. O conteúdo de um `TextBlock` é designado pelo objeto `TextElement`. Um objeto `ElementFormat` manipula a formatação do texto. A classe ElementFormat define propriedades como alinhamento da linha de base, ajuste de espaço, tracking, rotação de texto, além de tamanho, cor e caixa de fonte. Também inclui uma `FontDescription`, abordada em detalhes em "Trabalho com fontes" na página 420.

Uso do objeto ElementFormat

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O construtor do objeto `ElementFormat` adota qualquer item de uma lista longa de parâmetros opcionais, incluindo uma `FontDescription`. Também é possível definir essas propriedades fora do construtor. O exemplo a seguir mostra o relacionamento de vários objetos na definição e exibição de uma linha de texto simples.

```
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);
        }
    }
}
```

Cor e transparência (alfa) da fonte

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `color` do objeto `ElementFormat` define a cor da fonte. O valor é um inteiro que representa os componentes RGB da cor, por exemplo, 0xFF0000 para vermelho e 0x00FF00 para verde. O padrão é preto (0x000000).

A propriedade `alpha` define o valor da transferência alfa para um elemento (`TextElement` e `GraphicElement`). Os valores podem variar de 0 (totalmente transparente) a 1 (totalmente opaco, que é a definição padrão). Elementos com `alpha` igual a 0 são invisíveis, porém ativos. Esse valor é multiplicado por quaisquer valores alfa herdados, tornando o elemento mais transparente.

```
var ef:ElementFormat = new ElementFormat();
ef.alpha = 0.8;
ef.color = 0x999999;
```

Alinhamento e deslocamento de linha de base

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A fonte e o tamanho do texto maior em uma linha determina sua linha de base dominante. É possível substituir esses valores definindo `TextBlock.baselineFontDescription` e `TextBlock.baselineFontSize`. É possível alinhar a linha de base dominante com uma das diversas linhas de base dentro do texto. Essas linhas de base incluem as linhas ascendente e descendente ou as partes superior, inferior e central do ideograma.

A. *Ascendente* ***B.*** *Linha de base* ***C.*** *Descendente* ***D.*** *Altura do x*

No objeto `ElementFormat`, três propriedades determinam as características da linha de base e do alinhamento. A propriedade `alignmentBaseline` define a linha de base principal de um `TextElement` ou `GraphicElement`. Essa linha de base é a linha "de encaixe" do elemento, e a base de linha dominante de todo o texto se alinha em relação à sua posição.

A propriedade `dominantBaseline` especifica quais linhas de base do elemento usar, o que determina a posição vertical do elemento na linha. O valor padrão é `TextBaseline.ROMAN`, mas também pode ser definido de modo que `IDEOGRAPHIC_TOP` ou `IDEOGRAPHIC_BOTTOM` seja a linha de base dominante.

A propriedade `baselineShift` move a linha de base por um número de pixels definidos no eixo y. No texto normal (não girado), um valor positivo move a linha de base para baixo e um valor negativo, para cima.

Caixa tipográfica

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `TypographicCase` de `ElementFormat` especifica a caixa de texto como maiúscula, minúscula ou versalete.

```
var ef_Upper:ElementFormat = new ElementFormat();
ef_Upper.typographicCase = TypographicCase.UPPERCASE;

var ef_SmallCaps:ElementFormat = new ElementFormat();
ef_SmallCaps.typographicCase = TypographicCase.SMALL_CAPS;
```

Giro de texto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível girar um bloco de texto ou os glifos dentro de um segmento de texto em incrementos de 90°. A classe TextRotation define as seguintes constantes para definir o giro de bloco de texto e glifo:

Constante	Valor	Descrição
AUTO	“auto”	Especifica um giro de 90° no sentido anti-horário. Usada, normalmente, com a fonte asiática vertical para girar apenas glifos que exigem rotação.
ROTATE_0	“rotate_0”	Não especifica nenhuma rotação.
ROTATE_180	“rotate_180”	Especifica um giro de 180°.
ROTATE_270	“rotate_270”	Especifica um giro de 270°.
ROTATE_90	“rotate_90”	Especifica um giro de 90° no sentido horário.

Para girar as linhas de texto em um bloco de texto, defina a propriedade `TextBlock.lineRotation` antes de chamar o método `TextBlock.createTextLine()` para criar a linha de texto.

Para girar os glifos em um bloco de texto ou segmento, defina a propriedade `ElementFormat.textRotation` como o número de graus que o glifo deve ser girado. Um glifo é a forma que compõe um caractere ou uma parte de um caractere composto de vários glifos. A letra "a" e o ponto em um "i", por exemplo, são glifos.

O giro dos glifos é importante em alguns idiomas asiáticos em qual se quer girar as linhas para vertical, sem girar os caracteres dentro das linhas. Para obter mais informações sobre o giro de texto asiático, consulte "Justificação de texto do leste asiático" na página 424.

Este é um exemplo de giro de um bloco de texto e dos glifos internos, que pode ser aplicativo a um texto asiático. O exemplo também usa uma fonte japonesa:

```
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);
        }
    }
}
```

Bloqueio e clonagem de ElementFormat

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Quando um objeto `ElementFormat` for atribuído a qualquer tipo de `ContentElement`, sua propriedade `locked` será automaticamente definida como `true`. A tentativa para modificar um objeto `ElementFormat` bloqueado emite um `IllegalOperationError`. A prática recomendada é definir completamente tal objeto antes de atribuí-lo a uma instância do `TextElement`.

Para modificar uma instância do `ElementFormat` existente, primeiro verifique sua propriedade `locked`. Se estiver definida como `true`, use o método`clone()` para criar uma cópia desbloqueada do objeto. As propriedades desse objeto desbloqueado podem ser alteradas e é possível atribuí-lo à instância do `TextElement`. Quaisquer linhas criadas a partir de então terão a nova formatação. As linhas desse mesmo objeto criadas anteriormente, e usando o formato antigo, permanecerão inalteradas.

```
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);
        }
    }
}
```

Trabalho com fontes

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O objeto `FontDescription` é usado em conjunto com `ElementFormat` para identificar uma face de fonte e definir algumas de suas características. Essas características incluem o nome da fonte, peso, postura, renderização e como localizar a fonte (de dispositivo versus incorporada).

Nota: *O FTE não oferece suporte a fontes de Tipo 1 ou fontes de bitmap, como Tipo 3, ATC, sfnt-wrapped CID ou Naked CID.*

Definição das características da fonte (objeto FontDescription)

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `fontName` do objeto `FontDescription` pode ser um nome individual ou uma lista separada por vírgulas de nomes. Por exemplo, em uma lista como "Arial, Helvetica, _sans", o mecanismo de texto procura, primeiro, "Arial", em seguida, "Helvetica" e, finalmente, "_sans" se não for possível localizar as duas primeiras fontes. O conjunto de nomes de fontes inclui três nomes de fontes de dispositivo genéricos "_sans", "_serif" e "_typewriter". Eles mapeiam a fontes de dispositivos específicos, dependendo do sistema de reprodução. É prática recomendada especificar o nome padrão, por exemplo, os especificados em todas as descrições de fontes que usam fontes de dispositivo. Se nenhum `fontName` for especificado, "_serif" será usado como padrão.

A propriedade `fontPosture` pode ser definida como padrão (`FontPosture.NORMAL`) ou como itálico (`FontPosture.ITALIC`). A propriedade `fontWeight` pode ser definida como padrão (`FontWeight.NORMAL`) ou como negrito (`FontWeight.BOLD`).

```
var fd1:FontDescription = new FontDescription();
fd1.fontName = "Arial, Helvetica, _sans";
fd1.fontPosture = FontPosture.NORMAL;
fd1.fontWeight = FontWeight.BOLD;
```

Fontes incorporadas versus de dispositivo

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `fontLookup` do objeto `FontDescription` especifica se o mecanismo de texto procura uma fonte de dispositivo ou fonte incorporada para renderizar o texto. Se uma fonte de dispositivo (`FontLookup.DEVICE`) for especificada, o tempo de execução procurará a fonte no sistema de reprodução. A especificação de uma fonte incorporada (`FontLookup.EMBEDDED_CFF`) faz com que o tempo de execução procure uma fonte incorporada com o nome especificado no arquivo SWF. Somente fontes CFF (Formato compacto de fonte) funcionam com essa configuração. Se a fonte especificada não for localizada, uma fonte do dispositivo de fallback será usada.

As fontes de dispositivo resultam em um tamanho de arquivo SWF menor. As fontes incorporadas concedem maior fidelidade entre plataformas.

```
var fd1:FontDescription = new FontDescription();
fd1.fontLookup = FontLookup.EMBEDDED_CFF;
fd1.fontName = "Garamond, _serif";
```

Modo de renderização e instrução

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A renderização de CFF (Formato compacto de fonte) está disponível a partir do Flash Player 10 e do Adobe AIR 1.5. Esse tipo de renderização de fonte torna o texto mais legível e permite maior qualidade de exibição para fontes em tamanhos menores. Essa configuração aplica-se apenas a fontes incorporadas. `FontDescription` é a definição padrão dessa definição (`RenderingMode.CFF`) para a propriedade `renderingMode` . É possível definir essa propriedade como `RenderingMode.NORMAL` a fim de corresponder o tipo de renderização usado pelo Flash Player 7 ou versões anteriores.

Quando a renderização de CFF for selecionada, uma segunda propriedade, `cffHinting`, controlará como as hastes horizontais se ajustam à grade de subpixels. O valor padrão, `CFFHinting.HORIZONTAL_STEM`, usa instrução de CFF. Definir essa propriedade como `CFFHinting.NONE` remove a instrução, que é apropriada para animação ou para tamanhos de fontes maiores.

```
var fd1:FontDescription = new FontDescription();
fd1.renderingMode = RenderingMode.CFF;
fd1.cffHinting = CFFHinting.HORIZONTAL_STEM;
```

Bloqueio e clonagem de FontDescription

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Quando um objeto `FontDescription` estiver atribuído a um `ElementFormat`, sua propriedade `locked` será automaticamente definida como `true`. A tentativa para modificar um objeto `FontDescription` bloqueado emitiu um `IllegalOperationError`. A prática recomendada é definir completamente tal objeto antes de atribuí-lo a uma instância do `ElementFormat`.

Para modificar uma instância da `FontDescription` existente, primeiro verifique sua propriedade `locked`. Se estiver definida como `true`, use o método`clone()` para criar uma cópia desbloqueada do objeto. As propriedades desse objeto desbloqueado podem ser alteradas e é possível atribuí-lo à instância do `ElementFormat`. Quaisquer linhas criadas a partir desse `TextElement` terão a nova formatação. As linhas desse mesmo objeto criadas anteriormente permanecerão inalteradas.

```
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);
        }
    }
}
```

Controle de texto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O FTE proporcional um novo conjunto de controles de formatação de texto para manipular a justificação e o espaçamento de caracteres (kerning e tracking). Há também propriedades para controlar o modo como as linhas são quebradas e para definir as paradas de tabulação dentro das linhas.

Justificação do texto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A justificação de texto deixa todas as linhas em um parágrafo com o mesmo comprimento ajustando o espaçamento entre palavras e, algumas vezes, entre letras. O efeito é alinhar texto em ambos os lados, ao passo que o espaçamento entre palavras e letras varia. As colunas de texto nos jornais e revistas são frequentemente justificadas.

A propriedade `lineJustfication`, na classe SpaceJustifier, permite o controle da justificação das linhas em um bloco de texto. A classe LineJustification define as constantes que podem ser usadas para especificar uma opção de justificação: `ALL_BUT_LAST` justifica tudo, menos a última linha do texto; `ALL_INCLUDING_LAST` justifica todo o texto, inclusive a última linha; `UNJUSTIFIED`, que é o padrão, deixa o texto sem justificação.

Para justificar o texto, defina a propriedade `lineJustification` para uma instância da classe SpaceJustifier e atribua essa instância à propriedade `textJustifier` de uma instância do TextBlock. O exemplo a seguir cria um parágrafo em que todas as linhas do texto, menos a última, estão justificadas.

```
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);
            }
        }
    }
}
```

Para variar o espaçamento entre letras e palavras, defina a propriedade `SpaceJustifier.letterspacing` como `true`. A ativação do espaçamento entre letras reduz a ocorrência de lacunas de má aparência entre as palavras, o que pode ocorrer algumas vezes com a justificação simples.

Justificação de texto do leste asiático

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A justificação de texto do leste asiático exige considerações adicionais. Ele pode ser escrito do alto para baixo e alguns caracteres, conhecidos como kinsoku, não podem aparecer no início ou no fim de uma linha. A classe JustificationStyle define as seguintes constantes, que especificam as opções de manipulação desses caracteres. `PRIORITIZE_LEAST_ADJUSTMENT` baseia a justificação na expansão ou compressão da linha, dependendo do que produz o resultado mais desejável. `PUSH_IN_KINSOKU` baseará a justificação na compressão do kinsoku no final da linha ou na sua expansão, caso não haja nenhum kinsoku ou se o espaço não for suficiente.

PUSH_OUT_ONLY baseia a justificação na expansão da linha. Para criar um bloco de texto asiático vertical, defina a propriedade TextBlock.lineRotation para TextRotation.ROTATE_90 e defina a propriedade ElementFormat.textRotation para TextRotation.AUTO, que é o padrão. A definição da propriedade textRotation para AUTO faz com que os glifos no texto permaneçam na vertical em vez de virarem lateralmente quando a linha é girada. A configuração AUTO gira glifos, apenas de largura e amplitude completas, 90° no sentido anti-horário, conforme determinado pelas propriedades do Unicode do glifo. O exemplo a seguir exibe um bloco vertical de texto japonês e o justifica usando a opção PUSH_IN_KINSOKU.

```
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
```

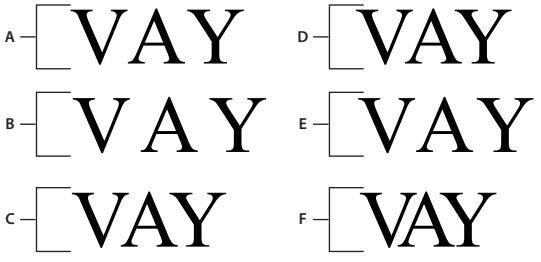
```
                font.fontName = String.fromCharCode(0x5C0F, 0x585A, 0x660E, 0x671D) + " Pro R";
            else
                font.fontName = "Kozuka Mincho Pro R";
            textBlock.content = new TextElement(Japanese_txt, format);
            var previousLine:TextLine = null;

            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 e tracking

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O kerning e o tracking afetam a distância entre os pares adjacentes de caracteres em um bloco de texto. O kerning controla como os pares de caracteres "se ajustam" entre si, por exemplo, os pares "WA" ou "Va". O kerning está definido no objeto `ElementFormat`. Está ativado por padrão (`Kerning.ON`) e pode ser definido como OFF ou AUTO; nesse caso, o kerning será aplicado entre os caracteres apenas se não forem Kanji, Hiragana ou Katakana.

O tracking adiciona ou subtrai um conjunto de números de pixels entre todos os caracteres em um bloco de texto e é definido também no objeto `ElementFormat`. Trabalha com ambas as fontes, incorporadas e de dispositivo. O FTE suporta duas propriedades de tracking: `trackingLeft`, que adiciona/subtrai pixels do lado esquerdo de um caractere, e `trackingRight`, que adiciona/subtrai do lado direito. Se o kerning estiver sendo usado, o valor de tracking será adicionado ou subtraído dos valores do kerning de cada par de caracteres.



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;
```

Quebras de linhas para texto disposto

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A propriedade `breakOpportunity` do objeto `ElementFormat` determina que caracteres podem ser usado para quebra de linha quando o texto está disposto em várias linhas. O padrão, `BreakOpportunity.AUTO`, usa as propriedades padrão do Unicode, como a quebra entre palavras ou nos hífens. O uso de `BreakOpportunity.ALL` permite que qualquer caractere seja tratado como uma oportunidade de quebra de linha, o que é útil para a criação de efeitos como texto ao longo de um caminho.

```
var ef:ElementFormat = new ElementFormat();
ef.breakOpportunity = BreakOpportunity.ALL;
```

Paradas de tabulação

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Para definir as paradas de tabulação em um bloco de texto, defina as paradas de tabulação criando instâncias da classe TabStop. Os parâmetros para o construtor `TabStop()` especifica como o texto se alinha à parada de tabulação. Esses parâmetros especificam a posição da parada de tabulação e, no caso do alinhamento decimal, o valor em que alinhar, expresso como uma sequência de caracteres. Normalmente, esse valor é um ponto decimal, mas também pode ser uma vírgula, um cifrão ou o símbolo do Yen ou Euro, por exemplo. A linha de código a seguir cria uma parada de tabulação denominada tab1.

```
var tab1:TabStop = new TabStop(TabAlignment.DECIMAL, 50, ".");
```

Depois da criação das paradas de tabulação para um bloco de texto, atribua-as à propriedade `tabStops` de uma instância do TextBlock. Como a propriedade `tabStops` requer um Vetor, crie primeiro o Vetor e, depois, adicione as paradas de tabulação a ele. O Vetor permite que você designe um conjunto de paradas de tabulação ao bloco de texto. O exemplo a seguir cria uma instância do `Vector<TabStop>` e inclui um conjunto de objetos TabStop nele. Em seguida, ele atribui as paradas de tabulação à propriedade `tabStops` de uma instância do TextBlock.

```
var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
tabStops.push(tab1, tab2, tab3, tab4);
textBlock.tabStops = tabStops
```

Para obter mais informações sobre Vetores, consulte “Trabalho com matrizes” na página 25.

O exemplo a seguir mostra o efeito de cada opção de alinhamento TabStop.

```
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;
            }
        }
    }
}
```

Exemplo de Flash Text Engine: layout das notícias

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Este exemplo de programação mostra o Flash Text Engine em uso organizando uma página simples de jornal. A página inclui um título grande, um subtítulo e uma seção de corpo com várias colunas.

Em primeiro lugar, crie um arquivo FLA e acrescente o código a seguir ao quadro #2 da camada padrão:

```
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();
```

StoryLayout.as é o script do controlador desse exemplo. Ele define o conteúdo, lê as informações de estilo de uma folha de estilos e atribui tais estilos aos objetos ElementFormat. Em seguida, cria o título, subtítulo e elementos de texto de várias colunas.

```
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 é usado como uma classe base para criação de blocos de texto. Inclui também funções utilitárias para alteração do tamanho e caixa da fonte.

```
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 estende a classe FormattedTextBlock e é usado para criação de títulos. Inclui uma função para ajuste de texto dentro do espaço definido na página.

```
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 manipula a formatação de texto dentro de um design de várias colunas. Demonstra o uso flexível de um objeto TextBlock como uma fábrica para criação, formatação e posicionamento de linhas de texto.

```
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);
        }

    }
}
```

Capítulo 23: Uso da Text Layout Framework

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Visão geral do Text Layout Framework

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A Text Layout Framework (TLF) é uma biblioteca ActionScript extensível. A TLF é construída sobre o mecanismo de texto no Adobe® Flash® Player 10 e no Adobe® AIR® 1.5. A TLF oferece recursos tipográficos e de layout de texto avançados para proporcionar uma tipografia inovadora na Web. A estrutura pode ser usada com o Adobe® Flex® ou com o Adobe® Flash® Professional. Os desenvolvedores podem usar ou estender componentes existentes ou podem usar a estrutura para criar seus próprios componentes de texto.

A TLF inclui os seguintes recursos:

- Texto bidirecional, texto vertical e mais de 30 scripts de escrita, inclusive árabe, hebraico, chinês, japonês, coreano, tailandês, laosiano, vietnamita, entre outros
- Seleção, edição e fluxo de texto em várias colunas e contêineres vinculados
- Texto vertical, Tate-Chu-Yoko (horizontal em texto vertical) e justificador para tipografia do leste asiático
- Controles tipográficos avançados, inclusive kerning, ligaturas, caso tipográfico, tipo de caixa de dígito, largura de dígito e hífens facultativos
- Recortar, copiar, colar, desfazer e gestos padrão do teclado e do mouse para edição
- APIs avançadas do desenvolvedor para manipular conteúdo de texto, layout e marcação e criar componentes de texto personalizados
- Suporte robusto de lista com marcadores personalizados e formatos de numeração
- Imagens alinhadas e regras de posicionamento

A TLF é uma biblioteca do ActionScript 3.0 construída com base no Flash Text Engine (FTE), lançado no Flash Player 10. O FTE pode ser acessado por meio do pacote `flash.text.engine` que é parte da API (Interface de programação de aplicativo) do Flash Player 10.

A API do Flash Player, no entanto, fornece bastante acesso de baixo nível ao mecanismo de texto, o que significa que algumas tarefas podem exigir uma quantidade relativamente grande de código. A TLF encapsula o código de baixo nível em APIs mais simples. A TLF também oferece uma arquitetura conceitual que organiza os elementos básicos definidos pelo FTE em um sistema mais fácil de usar.

Diferentemente do FTE, a TLF não é incorporada ao Flash Player. Em vez disso, trata-se de uma biblioteca de componentes independente escrita inteiramente no ActionScript 3.0. Como a estrutura é extensível, ela pode ser personalizada para ambientes específicos. Tanto o Flash Professional quanto o Flex SDK incluem componentes baseados na estrutura da TLF.

Mais tópicos da Ajuda

"Fluxo" do aplicativo de marcação TLF

Suporte a script complexo

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A TLF oferece suporte a script complexo. O suporte a script complexo inclui a capacidade de exibir e editar scripts da direita para a esquerda. A TLF também oferece a capacidade de exibir e editar uma mistura de scripts da esquerda para a direita e da direita para a esquerda, como árabe e hebraico. A estrutura oferece suporte não apenas para o layout de texto vertical para chinês, japonês e coreano, como também para tate-chuu-yoko (elementos TCY). Os elementos TCY são blocos de texto horizontal integrados em séries verticais de texto. Há suporte para os seguintes scripts:

- Latim (inglês, espanhol, francês, vietnamita e assim por diante)
- Grego, cirílico, armênio, georgiano e etíope
- Árabe e hebraico
- Ideogramas Han e Kana (chinês, japonês e coreano) e Hangul Johab (coreano)
- Tailandês, laosiano e khmer
- Devanagari, Bengali, Gurmukhi, Malaio, Télego, Tâmil, Gujarati, Oriya, Kannada e Tibetano
- Tifinagh, Yi, Cherokee, Silábico canadense, Deseret, Shaviano, Vai, Tagalo, Hanunoo, Buhid e Tagbanwa

Uso da Text Layout Framework no Flash Professional e no Flex

Você pode usar as classes TLF diretamente para criar componentes personalizados no Flash. Além disso, o Flash Professional CS5 fornece uma nova classe, fl.text.TLFTextField, que encapsula a funcionalidade da TLF. Use a classe TLFTextField para criar campos de texto no ActionScript que usam os recursos de exibição de texto avançados da TLF. Crie um objeto de TLFTextField do mesmo modo que você cria um campo de texto com a classe TextField. Em seguida, use a propriedade `textFlow` para atribuir a formatação avançada das classes TLF.

Você também pode usar o Flash Professional para criar a ocorrência de TLFTextField no palco usando a ferramenta de texto. Em seguida, pode usar o ActionScript para controlar a formatação e o layout do conteúdo do campo de texto usando classes da TLF. Para obter mais informações, consulte TLFTextField na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Se estiver trabalhando no Flex, use as classes da TLF. Para obter mais informações, consulte "Uso da Text Layout Framework" na página 439.

Uso da Text Layout Framework

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Se estiver trabalhando no Flex ou estiver criando componentes de texto personalizados, use as classes da TLF. A TLF é uma biblioteca do ActionScript 3.0 contida inteiramente na biblioteca textLayout.swc. A biblioteca TLF contém quase cem classes e interfaces do ActionScript 3.0 organizadas em dez pacotes. Esses pacotes são subpacotes do pacote flashx.textLayout.

Classes da Text Layout Framework

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

As classes da TLF podem ser agrupadas em três categorias:

- Estrutura de dados e formatação de classes
- Classes de renderização
- Classes de interação com o usuário

Estrutura de dados e formatação de classes

Os seguintes pacotes contêm as estruturas de dados e as classes de formatação da TLF:

- flashx.textLayout.elements
- flashx.textLayout.formats
- flashx.textLayout.conversion

A estrutura de dados principal da TLF é a hierarquia de fluxo de texto, definida no pacote de elementos. Dentro dessa estrutura, é possível atribuir estilos e atributos a execuções de texto com o pacote de formatos. Você também pode controlar como o texto é importado para a estrutura de cados, e dela exportado, com o pacote de conversão.

Classes de renderização

Os seguintes pacotes contêm as classes de renderização da TLF:

- flashx.textLayout.factory
- flashx.textLayout.container
- flashx.textLayout.compose

As classes nestes pacotes facilitam a renderização de texto para exibição pelo Flash Player. O pacote da fábrica fornece um modo simples para exibir texto estático. O pacote do contêiner inclui classes e interfaces que definem contêineres de exibição de texto dinâmico. O pacote composto define técnicas para posicionamento e exibição de texto dinâmico nos contêineres.

Classes de interação com o usuário

Os seguintes pacotes contêm as classes de interação com o usuário da TLF:

- flashx.textLayout.edit
- flashx.textLayout.operations
- flashx.textLayout.events

Os pacotes de edição e operações definem classes que podem ser usadas para permitir a edição do texto armazenado nas estruturas de dados. O pacote de eventos contém classes para tratamento de eventos.

Etapas gerais para criação de texto com a Text Layout Framework

As seguintes etapas descrevem o processo geral para criação de texto com o Text Layout Format:

1. Importe texto formatado para as estruturas de dados da TLF. Para obter mais informações, consulte “Estruturação de texto com a TLF” na página 444 e “Formatação de texto com a TLF” na página 448.
2. Crie um ou mais contêineres de objeto de exibição vinculados para o texto. Para obter mais informações, consulte “Gerenciamento de contêineres de texto com a TLF” na página 450.

3 Associe o texto nas estruturas de dados aos contêineres e defina opções de edição e rolagem. Para obter mais informações, consulte "Ativação da seleção de texto, edição e recurso desfazer com a TLF" na página 451.

4 Crie um manipulador de evento para fazer o refluxo do texto em resposta a eventos de redimensionamento (ou outros). Para obter mais informações, consulte "Tratamento de eventos com a TLF" na página 451.

Exemplo da Text Layout Framework: layout das notícias

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O exemplo a seguir demonstra o uso da TLF para fazer o layout de uma simples página de jornal. A página inclui um título grande, um subtítulo e uma seção de corpo com várias colunas:

```
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();
        }
    }
}
```

A classe TLFNewsLayout usa dois contêineres de texto. Um contêiner exibe um título e subtítulo e o outro exibe o texto do corpo em três colunas. Por uma questão de simplicidade, o texto é codificado no exemplo como texto de marcação TLF. A variável `headlineMarkup` contém o título e o subtítulo, e a variável `bodyMarkup` contém o corpo do texto. Para obter mais informações sobre a marcação da TLF, consulte "Estruturação de texto com a TLF" na página 444.

Após alguma inicialização, a função `onAddedToStage()` importa o texto do título para o objeto TextFlow, que é a principal estrutura de dados da TLF:

```
hTextFlow = TextConverter.importToFlow(headlineMarkup, TextConverter.TEXT_LAYOUT_FORMAT);
```

Em seguida, um objeto Sprite é criado para o contêiner e um controlador é criado e associado ao contêiner:

```
headContainer = new Sprite();
headlineController = new ContainerController(headContainer);
```

O controlador é inicializado para definir formatação e rolagem, entre outras opções. O controlador contém geometria que define os limites do contêiner no qual o texto flui. Um objeto TextLayoutFormat contém as opções de formatação:

```
hContainerFormat = new TextLayoutFormat();
```

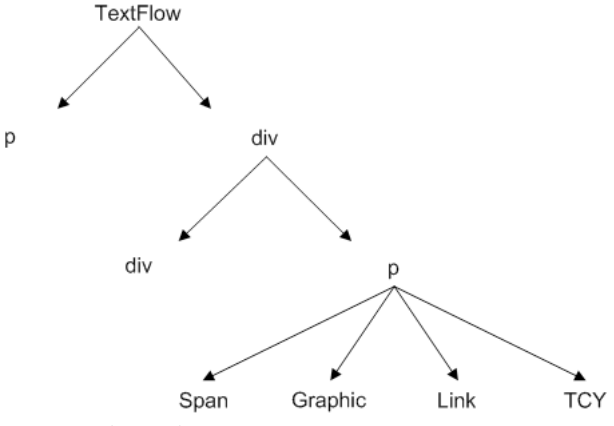
O controlador é atribuído ao compositor de fluxo e a função adiciona o contêiner à lista de exibição. A composição e a exibição reais dos contêineres são adiados para o método `resizeHandler()`. A mesma sequência de etapas é executada para inicializar o objeto TextFlow do corpo.

O método `resizeHandler()` mede o espaço disponível para renderizar os contêineres e dimensioná-los de acordo. Uma chamada ao método `compose()` permite calcular a altura adequada do contêiner de título. O método `resizeHandler()` pode então colocar e exibir o contêiner de título com o método `updateAllControllers()`. Por fim, o método `resizeHandler()` usa o tamanho do contêiner de título para determinar a colocação do contêiner de texto do corpo.

Estruturação de texto com a TLF

A TLF usa uma árvore hierárquica para representar o texto. Cada nó na árvore é a instância de uma classe definida no pacote de elementos. Por exemplo, o nó raiz da árvore sempre é uma instância da classe TextFlow. A classe TextFlow representa uma matéria inteira de texto. Uma história é uma coleção de texto e outros elementos tratada como uma unidade única, ou fluxo. Um único artigo história pode exigir mais do que uma coluna ou contêiner de texto para ser exibida.

Além do nó raiz, os elementos remanescentes baseiam-se livremente nos elementos XHTML. O diagrama a seguir mostra a hierarquia da estrutura:



Hierarquia de TextFlow

Marcação da Text Layout Framework

Compreender a estrutura da TLF também é útil ao lidar com a marcação TLF. A marcação TLF é uma representação de texto que faz parte da TLF. Embora a estrutura também ofereça suporte a outros formatos XML, a marcação TLF é exclusiva pelo fato de se basear especificamente na estrutura da hierarquia TextFlow. Se você exportar o XML de um TextFlow usando esse formato de markup, o XML será exportado com essa hierarquia intacta.

A marcação TLF oferece a representação da mais alta fidelidade em uma hierarquia TextFlow. A linguagem da marcação fornece tags para cada um dos elementos básicos da hierarquia TextFlow, além de fornecer atributos para todas as propriedades de formatação disponíveis na classe TextLayoutFormat.

A tabela a seguir contém as tags que podem ser usadas na TLF Markup.

Elemento	Descrição	Filhos	Classe
textflow	O elemento raiz da marcação.	div, p	TextFlow
div	Uma divisão em um TextFlow. Pode conter um grupo de parágrafos.	div, list, p	DivElement
p	Um parágrafo.	a, tcy, span, img, tab, br, g	ParagraphElement
a	Um link.	tcy, span, img, tab, br, g	LinkElement
tcy	Uma série de texto horizontal (usada em um TextFlow vertical).	a, span, img, tab, br, g	TCYElement
span	Uma série de texto em um parágrafo.		SpanElement
img	Uma imagem em um parágrafo.		InlineGraphicElement
tab	Um caractere de tabulação.		TabElement
br	Uma caractere de quebra. Usado para terminar uma linha em um parágrafo. O texto continua na próxima linha, mas permanece no mesmo parágrafo.		BreakElement
linkNormalFormat	Define os atributos de formatação usados para links no estado normal.	TextLayoutFormat	TextLayoutFormat
linkActiveFormat	Define os atributos de formatação usados para links no estado ativo, quando o mouse está sobre um link.	TextLayoutFormat	TextLayoutFormat
linkHoverFormat	Define os atributos de formatação usados para links no estado suspenso, quando o mouse está dentro dos limites (rolando sobre) de um link.	TextLayoutFormat	TextLayoutFormat
li	Um elemento de item da lista. Deve estar dentro de um elemento da lista.	div, li, list, p	ListItemElement
list	Uma lista. As listas podem ser aninhadas, ou colocadas adjacentes entre si. Diferentes esquemas de rotulagem ou de numeração podem ser aplicados aos itens da lista.	div, li, list, p	ListElement
g	Um elemento do grupo. Usado para agrupar elementos em um parágrafo. Permite aninhar elementos abaixo do nível de parágrafo.	a, tcy, span, img, tab, br, g	SubParagraphGroupElement

Mais tópicos da Ajuda

Markup de listas TLF 2.0

TLF 2.0 SubParagraphGroupElements e typeName

Usando listas enumeradas e com marcadores

É possível usar as classes ListElement e ListItemElement para incluir listas com marcadores nos controles do seu texto. As listas com marcadores podem ser aninhadas e personalizadas para usar marcadores (ou sinalizadores) diferentes e numeração automática, assim como numeração por destaque.

Para criar listas de seus fluxos de texto, use a tag `<list>`. Use tags `<li>` dentro da tag `<list>` para cada item de lista na lista. É possível personalizar a aparência dos marcadores usando a classe ListMarkerFormat.

O exemplo a seguir cria listas simples:

```
<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>
```

É possível aninhar listas dentro de outras, como mostra o exemplo a seguir:

```
<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>
```

Para personalizar o tipo de marcador na lista, use a propriedade `listStyleType` do ListElement. Esta propriedade pode ser qualquer valor definido pela classe ListStyleType (como `check`, `circle`, `decimal` e `box`). O exemplo a seguir cria listas com vários tipos de marcadores e um incremento de contador personalizado:

```
<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>
```

Use a classe ListMarkerFormat para definir o contador. Além de definir o incremento de um contador, também é possível personalizá-lo, redefinindo-o com a propriedade `counterReset`.

É possível ainda personalizar a aparência dos marcadores em sua lista usando as propriedades `beforeContent` e `afterContent` do ListMarkerFormat. Estas propriedades se aplicam ao conteúdo que aparece antes e depois do conteúdo do marcador.

O exemplo a seguir adiciona a sequência de caracteres “XX” antes do marcador, e a “YY” antes:

```
<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>
```

A propriedade `content` em si pode definir mais personalizações do formato do marcador. O exemplo a seguir exibe um marcador ordenado de numeral romano maiúsculo:

```
<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>
```

Como mostra o exemplo anterior, a propriedade `content` também pode inserir um sufixo: uma sequência de caracteres que aparece depois do marcador, mas antes do `afterContent`. Para inserir esta sequência de caracteres ao fornecer o conteúdo XML para o fluxo, envolva a sequência de caracteres nas entidades HTML `&quote;` em vez de aspas (`"<string>"`).

Mais tópicos da Ajuda

Markup de listas TLF 2.0

Usando deslocamento no TLF

Cada FlowElement é compatível com propriedades de deslocamento que você usa para controlar a posição da área do conteúdo de cada elemento, e o espaço entre as áreas do conteúdo.

A largura total de um elemento é a soma da largura do seu conteúdo, mais as propriedades `paddingLeft` e `paddingRight`. A altura total de um elemento é a soma da altura do seu conteúdo, mais as propriedades `paddingTop` e `paddingBottom`.

O deslocamento é o espaço entre a borda e o conteúdo. As propriedades de deslocamento são `paddingBottom`, `paddingTop`, `paddingLeft` e `paddingRight`. É possível aplicar pads ao objeto TestFlow e aos seguintes elementos subordinados (filhos):

- div
- img
- li
- list
- p

As propriedades de deslocamento não podem ser aplicadas aos elementos de distância.

O exemplo a seguir define as propriedades de deslocamento no TextFlow:

```
<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">
```

Os valores válidos para as propriedades de panorama são um número (em pixels), “auto”, ou “herdado”. O valor padrão é "auto", o que significa que é calculado automaticamente e definido para 0, para todos os elementos, exceto o ListElement. Para ListElements, “auto” é 0 exceto no lado de início da lista, onde o valor da propriedade `listAutoPadding` é usado. O valor padrão de `listAutoPadding` é 40, o que dá à lista um recuo padrão.

As propriedades de deslocamento não são herdadas por padrão. Os valores “auto” e “inherit” são constantes definidas pela classe FormatValue.

As propriedades de deslocamento podem ser valores negativos.

Mais tópicos da Ajuda

Alterações de deslocamento em TLF 2.0

Formatação de texto com a TLF

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O pacote flashx.textLayout.formats contém interfaces e classes que permitem atribuir formatos a qualquer FlowElement na árvore da hierarquia de fluxo de texto. Há dois modos para aplicação da formatação. Você pode atribuir um formato específico individualmente ou atribuir um grupo de formatos simultaneamente com um objeto de formatação especial.

A interface ITextLayoutFormat contém todos os formatos que podem ser aplicados a um FlowElement. Alguns formatos se aplicam a um parágrafo ou contêiner de texto inteiro, mas não se aplicam a caracteres individuais. Por exemplo, formatos como justificação e paradas de tabulação se aplicam a todos os parágrafos, mas não são aplicáveis a caracteres individuais.

Atribuição de formatos a um FlowElement com propriedades

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Você pode definir formatos individuais em FlowElement através de atribuição de propriedade. A classe FlowElement implementa a interface ITextLayoutFormat, de modo que qualquer subclasse da classe FlowElement também deve implementar essa interface.

Por exemplo, o código a seguir mostra como atribuir formatos individuais a uma ocorrência de ParagraphElement:

```
var p:ParagraphElement = new ParagraphElement();
p.fontSize = 18;
p.fontFamily = "Arial";
```

Atribuição de formatos a um FlowElement com a classe TextLayoutFormat

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

É possível aplicar formatos a um FlowElement com a classe TextLayoutFormat. Utilize esta classe para criar um objeto de formatação especial que contém todos os valores e formato que desejar. Isso viabilizará a designação desse objeto à propriedade `format` de qualquer objeto FlowElement. Tanto TextLayoutFormat quanto FlowElement implementam a interface ITextLayoutFormat. Essa organização garante que as duas classes contenham as mesmas propriedades de formato.

Para obter mais informações, consulte TextLayoutFormat na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Herança de formato

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Formatos são herdados através da hierarquia de fluxo de texto. Se você designar uma instância de TextLayoutFormat a uma instância do FlowElement com filhos, a estrutura iniciará um processo denominado *cascade*. Durante uma cascata, a estrutura examina recursivamente cada nó na hierarquia que herda do seu FlowElement. Então, ela determina se deve atribuir os valores herdados para cada propriedade de formatação. As seguintes regras se aplicam à réplica em cascata:

1. Os valores da propriedade são herdados apenas de um ancestral imediato (algumas vezes, denominado o pai).
2. Os valores da propriedade serão herdados apenas se uma propriedade não tiver ainda nenhum valor (ou seja, o valor estiver `undefined`).
3. Alguns atributos não herdarão valores indefinidos, a não ser que o valor do atributo esteja definido como "inherit" ou como a constante `flashx.textLayout.formats.FormatValue.INHERIT`.

Por exemplo, se você definir o valor `fontSize` no nível do TextFlow, a configuração se aplicará a todos os elementos no TextFlow. Em outras palavra, os valores replicam a hierarquia de fluxo de texto em cascata. Será possível, no entanto, substituir o valor em determinado elemento designando-lhe diretamente um novo valor. Como contraexemplo, se você definir o valor `backgroundColor` para o nível do TextFlow, os filhos do TextFlow não herdam esse valor. A propriedade `backgroundColor` é aquela que não herda de seu pai durante uma cascata. É possível sobrescrever esse comportamento definido à propriedade `backgroundColor` em cada filho para `flashx.textLayout.formats.FormatValue.INHERIT`.

Para obter mais informações, consulte TextLayoutFormat na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Importação e exportação de texto com a TLF

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A classe TextConverter no flashx.textLayout.conversion.* o pacote permite importar e exportar texto do TLF. Use essa classe se planeja carregar texto no tempo de execução em vez de compilar o texto no arquivo SWF. Você também pode usar essa classe para exportar o texto, que fica armazenado em uma ocorrência de TextFlow em um objeto String ou XML.

Tanto a importação quanto a exportação são procedimentos simples. Você chama o método `export()` ou `importToFlow()`, ambos partes da classe TextConverter. Ambos os métodos são estáticos, o que significa que você chama os métodos na classe TextConverter em vez de uma instância da classe.

As classes no pacote flashx.textLayout.conversion oferecem considerável flexibilidade por permitir a opção de armazenar o texto. Por exemplo, se você armazenar seu texto em um banco de dados, poderá importar o texto para a estrutura para fins de exibição. Em seguida, você pode usar as classes no pacote flashx.textLayout.edit para alterar o texto e exportar o texto alterado de volta para o banco de dados.

Para obter mais informações, consulte flashx.textLayout.conversion na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Gerenciamento de contêineres de texto com a TLF

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Depois que o texto é armazenado nas estruturas de dados da TLF, o Flash Player pode exibi-lo. O texto armazenado na hierarquia de fluxo precisa ser convertido em um formato que o Flash Player possa exibir. O TLF oferece duas maneiras de criar objetos de exibição a partir de um fluxo. A primeira, a abordagem mais simples, é adequada para exibição de texto estático. A segunda, a abordagem mais complicada, permite criar texto dinâmico que pode ser selecionado e editado. Em ambos os casos, o texto é finalmente convertido em instâncias da classe TextLine, que faz parte do pacote flash.text.engine.* no Flash Player 10.

Criação de texto estático

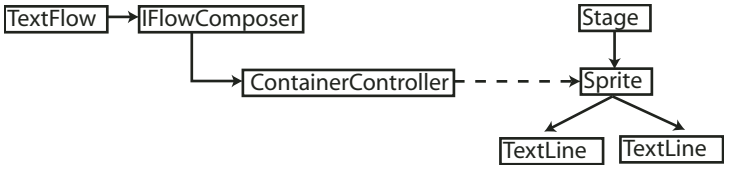
A abordagem simples utiliza a classe TextFlowTextLineFactory, que pode ser encontrada no pacote flashx.textLayout.factory. A vantagem dessa abordagem, além da sua simplicidade, é que ela tem uma superfície de memória menor em relação à abordagem do FlowComposer. Essa abordagem é aconselhável para o texto estático que o usuário não precisa editar, selecionar ou rolar.

Para obter mais informações, consulte TextFlowTextLineFactory na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Criação de texto dinâmico e de contêineres

Use um compositor de fluxo se desejar ter mais controle sobre a exibição do texto do que quando fornecido pelo TextFlowTextLineFactory. Por exemplo, com um compositor de fluxo, os usuários podem selecionar e editar o texto. Para obter mais informações, consulte "Ativação da seleção de texto, edição e recurso desfazer com a TLF" na página 451.

Um compositor de fluxo é uma ocorrência da classe StandardFlowComposer no pacote flashx.textLayout.compose. Um compositor de fluxo gerencia a conversão do TextFlow em ocorrências de TextLine, bem como o posicionamento dessas ocorrências de TextLine em um ou mais contêineres.



Um IFlowComposer tem zero ou mais ContainerControllers

Cada instância do TextFlow tem um objeto correspondente que implementa a interface IFlowComposer. Esse objeto IFlowComposer é acessível por meio da propriedade `TextFlow.flowComposer`. Você chama métodos definidos pela interface IFlowComposer usando esta propriedade. Estes métodos permitem associar o texto a um ou mais contêineres e preparar o texto para exibição em um contêiner.

Um contêiner é uma instância da classe Sprite, a qual, por sua vez, é uma subclasse da classe DisplayObjectContainer. Ambas essas classes são parte da API da lista de exibição do Flash Player. Um contêiner é uma forma mais avançada do retângulo delimitador usado na classe TextLineFactory. Como o retângulo delimitador, um contêiner define a área em que as instâncias do TextLine aparecerão. Diferentemente de um retângulo delimitador, um contêiner te um objeto "controlador" correspondente. O controlador gerencia a rolagem, composição, vinculação, formatação e tratamento de eventos de um contêiner ou conjunto de contêineres. Cada contêiner tem um objeto controlador correspondente, que é uma ocorrência da classe ContainerController no pacote flashx.textLayout.container.

Para exibir texto, crie um objeto controlador para gerenciar o contêiner e associá-lo ao compositor de texto. Depois de associar o contêiner, componha o texto antes que ele possa ser exibido. Da mesma forma, os contêineres têm dois estados: composição e exibição. A composição é o processo de conversão de texto da hierarquia de fluxo de texto nas instâncias do TextLine e o cálculo de se essas instâncias se ajustarão ao contêiner. A exibição é o processo de atualização da lista de exibição do Flash Player.

Para obter mais informações, consulte IFlowComposer, StandardFlowComposer e ContainerController na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Ativação da seleção de texto, edição e recurso desfazer com a TLF

Flash Player 9.0 e posterior, Adobe AIR 1.0 e posterior

A capacidade de selecionar ou editar texto é controlada no nível do fluxo de texto. Todas as instâncias da classe TextFlow tem um gerenciador de interação associado. Você pode acessar o gerenciador de interação de um objeto TextFlow através da propriedade `TextFlow.interactionManager` do objeto. Para ativar a seleção de texto, designe uma instância da classe SelectionManager à propriedade `interactionManager`. Para ativar a seleção e edição de texto, designe uma instância da classe EditManager em vez de uma instância da classe SelectionManager. Para ativar as operações de desfazer, crie uma instância da classe UndoManager e inclua-a como argumento ao chamar o construtor para EditManager. A classe UndoManager mantém um histórico das atividades de edição mais recentes do usuário e que o usuário desfaça ou refaça edições específicas. As três classes fazem parte do pacote de edição.

Para obter mais informações, consulte SelectionManager, EditManager e UndoManager na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Tratamento de eventos com a TLF

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Objetos TextFlow despacham eventos em muitas circunstâncias, que incluem:

- Quando o texto ou o layout são alterados
- Antes do início e após o fim de uma operação
- Quando o status de um objeto FlowElement é alterado

- Quando uma operação de composição é concluída

Para obter mais informações, consulte flashx.textLayout.events na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Mais tópicos da Ajuda

Eventos TLF FlowElement, LinkElement e EventMirrors

Posicionamento de imagens dentro do texto

Para posicionar InlineGraphicElement dentro do texto, use as seguintes propriedades:

- propriedade `float` da classe InlineGraphicElement
- propriedade `clearFloats` da FlowElement

A propriedade `float` controla o posicionamento do gráfico e o texto em torno deste. A propriedade `clearFloats` controla o posicionamento dos elementos do parágrafo em relação ao `float`.

Para controlar a localização de uma imagem dentro de um elemento de texto, use a propriedade `float`. O exemplo a seguir adiciona uma imagem a um parágrafo e a alinha à esquerda para que o texto seja disposto em torno da direita:

```
<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>
```

Os valores válidos para a propriedade `float` são “left”, “right”, “start”, “end” e “none”. A classe Float define estas constantes. O valor padrão é "none".

A propriedade `clearFloats` é útil nos casos em que você deseja ajustar a posição inicial dos parágrafos subsequentes, que normalmente envolve a imagem. Por exemplo, suponha que você tenha uma imagem que é maior do que o primeiro parágrafo. Para ter certeza que o segundo parágrafo inicia *depois* da imagem, defina a propriedade `clearFloats`.

O exemplo a seguir usa uma imagem mais alta do que o texto no primeiro parágrafo. Para fazer o segundo parágrafo iniciar após a imagem do bloco de texto, este exemplo define a propriedade `clearFloats` no segundo parágrafo para "end".

```
<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>
```

Os valores válidos para a propriedade `clearFloats` são “left”, “right”, “end”, “start”, “none” e “both”. A classe ClearFloats define estas constantes. Também é possível definir a propriedade `clearFloats` para “inherit”, que é uma constante definida pela classe FormatValue. O valor padrão é "none".

Mais tópicos da Ajuda

TLF Floats

Capítulo 24: Trabalho com som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript foi desenvolvido para aplicativos interativos de imersão, e um dos elementos muitas vezes negligenciado nesses aplicativos é o som. É possível adicionar efeitos de som a um video game, feedback de áudio para a interface do usuário de um aplicativo ou mesmo criar um programa que analisa arquivos MP3 carregados pela Internet, com som na base do aplicativo.

Você pode carregar os arquivos de áudio externos e trabalhar com o áudio que está incorporado em um SWF. Você poderá controlar o áudio, criar representações visuais das informações de som e capturar som de um microfone do usuário.

Mais tópicos da Ajuda

pacote flash.media

flash.events.SampleDataEvent

Noções básicas do trabalho com som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os computadores podem capturar e codificar áudio digital, que é a representação das informações de som no computador, bem como armazená-lo e recuperá-lo para reprodução pelos alto-falantes. É possível reproduzir som usando o Adobe® Flash® Player ou Adobe® AIR™ e o ActionScript.

Quando dados de som são convertidos em formato digital, eles têm várias características, como o volume e se o som é estéreo ou mono. Quando você reproduz um som no ActionScript, também pode ajustar essas características; por exemplo, você pode aumentar o som ou fazer parecer que ele está vindo de uma certa direção.

Para controlar um som no ActionScript, você precisa ter as informações do som carregadas no Flash Player ou no AIR. Há cinco maneiras de inserir dados de áudio no Flash Player ou AIR para que você possa trabalhar com eles no ActionScript.

- Carregue um arquivo de som externo, como um arquivo mp3 no SWF.
- Incorpore as informações de som diretamente no arquivo SWF durante sua criação.
- Capture áudio de um microfone conectado ao computador do usuário.
- Faça o fluxo de áudio a partir de um servidor.
- Gere e reproduza áudio dinamicamente.

Quando você carrega dados de som a partir de um arquivo de som externo, pode começar a reproduzir o início do arquivo enquanto o restante dos dados ainda estão sendo carregados.

Embora existam vários formatos de arquivo de som usados para codificar áudio digital, o ActionScript 3.0, o Flash Player e o AIR dão suporte a arquivos de som armazenados no formato mp3. Eles não podem carregar ou reproduzir arquivos de som diretamente em outros formatos, como WAV ou AIFF.

Enquanto estiver trabalhando com som no ActionScript, provavelmente você desejará trabalhar com várias classes do pacote flash.media. A classe Sound é a classe que você usa para obter acesso a informações de áudio carregando um arquivo de som ou atribuindo uma função a um evento que tira amostras de dados de som e, em seguida, inicia a reprodução. Depois que você começa a reproduzir um som, o Flash Player e o AIR lhe dão acesso a um objeto SoundChannel. Como um arquivo de áudio carregado só pode ser um dos vários sons que você reproduz no computador de um usuário, cada som específico reproduzido usa seu próprio objeto SoundChannel; a saída combinada de todos os objetos SoundChannel misturados na verdade consiste no que é reproduzido pelos alto-falantes do computador. Use esta ocorrência de SoundChannel para controlar as propriedades do som e interromper a reprodução. Por último, se você deseja controlar o áudio combinado, a classe SoundMixer lhe dá controle sobre a saída combinada.

Também é possível usar várias outras classes para executar tarefas mais específicas quando você estiver trabalhando com som no ActionScript; para obter mais informações sobre todas as classes relacionadas a som, consulte "Compreensão da arquitetura do som" na página 454.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes que você poderá encontrar:

Amplitude A distância de um ponto na forma de onda de som a partir da linha zero ou de equilíbrio.

Taxa de bits A quantidade de dados que é codificada ou transmitida em fluxo para cada segundo de um arquivo de som. No caso de arquivos mp3, a taxa de bits normalmente é informada em milhares de bits por segundo (kbps). Uma taxa de bits mais alta geralmente significa uma onda de som de melhor qualidade.

Armazenamento em buffer O recebimento e o armazenamento de dados de som antes de eles serem reproduzidos.

mp3 MPEG-1 Audio Layer 3, ou mp3, é um formato popular de compactação de som.

Panorâmica O posicionamento de um sinal de áudio entre os canais esquerdo e direito em um campo de som estéreo.

Pico O ponto mais alto em uma forma de onda.

Taxa de amostragem Define o número de amostras por segundo extraídas de um sinal de áudio analógico para criar um sinal digital. A taxa de amostragem de áudio de um CD padrão é de 44,1 kHz ou 44.100 amostras por segundo.

Streaming O processo de reproduzir as partes iniciais de um arquivo de som ou de vídeo enquanto as partes finais ainda estão sendo carregadas de um servidor.

Volume O volume de um som.

Forma de onda A forma de um gráfico das variadas amplitudes de um sinal de som ao longo do tempo.

Compreensão da arquitetura do som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Seus aplicativos podem carregar dados de som de cinco origens principais:

- Arquivos de som externos carregados em tempo de execução
- Recursos de som incorporados no arquivo SWF do aplicativo
- Dados de som de um microfone conectado ao sistema do usuário
- Dados de som transmitidos de um servidor de mídia remoto, como o Flash Media Server
- Dados de som gerados dinamicamente por meio do uso do manipulador de eventos `sampleData`

Dados de som podem ser carregados completamente antes de o som ser reproduzido ou podem ser transmitidos em fluxo, o que significa que o som é reproduzido enquanto ainda está sendo carregado.

As classes de som do ActionScript 3.0 dão suporte a arquivos de som armazenados no formato mp3. Elas não podem carregar ou reproduzir arquivos de som diretamente em outros formatos, como WAV ou AIFF. No entanto, a partir do Flash Player 9.0.115.0, é possível carregar e reproduzir arquivos de áudio AAC usando a classe NetStream. Esta técnica é a mesma utilizada para carregar e reproduzir conteúdo de vídeo. Para obter mais informações sobre ela, consulte "Trabalho com vídeo" na página 487.

Com o Adobe Flash Professional, é possível importar arquivos de som WAV ou AIFF e incorporá-los aos arquivos SWF do aplicativo no formato mp3. A ferramenta de autoria do Flash também permite compactar arquivos de som incorporados para reduzir o tamanho do arquivo, embora essa redução de tamanho represente perda da qualidade do som. Para obter mais informações, consulte "Importação de sons" em *Uso do Flash*.

A arquitetura de som do ActionScript 3.0 usa as seguintes classes no pacote flash.media.

Classe	**Descrição**
flash.media.Sound	A classe Sound manipula o carregamento do som, gerencia propriedades básicas de som e inicia uma reprodução de som.
flash.media.SoundChannel	Quando um aplicativo reproduz um objeto Sound, um novo objeto SoundChannel é criado para controlar a reprodução. O objeto SoundChannel controla o volume dos canais de reprodução direito e esquerdo do som. Cada som reproduzido tem seu próprio objeto SoundChannel.
flash.media.SoundLoaderContext	A classe SoundLoaderContext especifica o número de segundos de buffer a ser usado ao carregar um som e se o Flash Player ou o AIR deve procurar um arquivo de política no servidor quando carregar um arquivo. Um objeto SoundLoaderContext é usado como um parâmetro para o método `Sound.load()`.
flash.media.SoundMixer	A classe SoundMixer controla as propriedades de reprodução e de segurança relativas a todos os sons em um aplicativo. Em vigor, vários canais de som são misturados por meio de um objeto SoundMixer comum, portanto valores de propriedades no objeto SoundMixer afetarão todos os objetos SoundChannel em execução no momento.
flash.media.SoundTransform	A classe SoundTransform contém valores que controlam o volume e o panorama do som. Um objeto SoundTransform pode ser aplicado a um objeto SoundChannel individual, ao objeto SoundMixer global ou a um objeto Microphone, entre outros.
flash.media.ID3Info	Um objeto ID3Info contém propriedades que representam informações de metadados ID3 que normalmente são armazenados em arquivos de som mp3.
flash.media.Microphone	A classe Microphone representa um microfone ou outro dispositivo de entrada de som conectado ao computador do usuário. A entrada de áudio de um microfone pode ser roteada para alto-falantes locais ou enviada a um servidor remoto. O objeto Microphone controla o ganho, a taxa de amostragem e outras características de seu próprio fluxo de som.
flash.media.AudioPlaybackMode	A classe AudioPlaybackMode define constantes para a propriedade `audioPlaybackMode` da classe SoundMixer.

Cada som carregado e reproduzido precisa de sua própria ocorrência das classes Sound e SoundChannel. A saída das várias ocorrências de SoundChannel é então misturada pela classe global SoundMixer durante a reprodução.

As classes Sound, SoundChannel e SoundMixer não são usadas para dados de som obtidos de um microfone ou de um servidor de fluxo de mídia, como o Flash Media Server.

Carregamento de arquivos de som externos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cada ocorrência da classe Sound existe para carregar e disparar a reprodução de um recurso de som específico. Um aplicativo não pode reutilizar um objeto Sound para carregar mais de um som. Para carregar um novo recurso de som, ele deve criar um novo objeto Sound.

Se estiver carregando um arquivo de som pequeno, como um som de clique a ser conectado a um botão, o aplicativo poderá criar um novo Sound e fazer com que ele carregue automaticamente o arquivo de som, conforme mostrado a seguir:

```
var req:URLRequest = new URLRequest("click.mp3");
var s:Sound = new Sound(req);
```

O construtor `Sound()` aceita um objeto URLRequest como seu primeiro parâmetro. Quando um valor é fornecido ao parâmetro URLRequest, o novo objeto Sound começa a carregar o recurso de som especificado automaticamente.

Em todos, menos nos casos mais simples, o aplicativo deve verificar se ocorrem erros durante o carregamento do som. Por exemplo, se o som de clique for razoavelmente grande, ele talvez não esteja completamente carregado quando o usuário clicar no botão que dispara o som. A tentativa de reproduzir um som não carregado pode provocar um erro em tempo de execução. É mais seguro aguardar que o carregamento do som seja concluído antes de permitir que os usuários executem ações que podem iniciar a reprodução de sons.

Um objeto Sound despacha vários eventos diferentes durante o processo de carregamento do som. O aplicativo pode ouvir esses eventos para controlar o progresso de carregamento e verificar se o som foi completamente carregado antes da reprodução. A tabela a seguir lista os eventos que podem ser despachados por um objeto Sound.

Evento	Descrição
abertura (`Event.OPEN`)	Despachado imediatamente antes do início da operação de carregamento do som.
progresso (`ProgressEvent.PROGRESS`)	Despachado periodicamente durante o processo de carregamento do som quando os dados são recebidos do arquivo ou do fluxo.
id3 (`Event.ID3`)	Despachado quando dados ID3 estão disponíveis para um som mp3.
concluído (`Event.COMPLETE`)	Despachado quando todos os dados do recurso de som foram carregados.
ioError (`IOErrorEvent.IO_ERROR`)	Despachado quando um arquivo de som não pode ser localizado ou quando o processo de carregamento é interrompido antes dos dados serem recebidos.

O código a seguir ilustra como reproduzir um som após a conclusão do carregamento:

```
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();
}
```

Primeiro, o código de amostra cria um novo objeto Sound sem dar a ele um valor inicial para o parâmetro URLRequest. Em seguida, ele ouve o evento `Event.COMPLETE` do objeto Sound, o que faz com que o método `onSoundLoaded()` seja executado quando todos os dados do som estão carregados. Em seguida, ele chama o método `Sound.load()` com um novo valor de URLRequest para o arquivo de som.

O método `onSoundLoaded()` é executado quando o carregamento do som é concluído. A propriedade `target` do objeto Event é uma referência ao objeto Sound. A chamada do método `play()` do objeto Sound inicia a reprodução do som.

Monitoramento do processo de carregamento do som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Arquivos de som podem ser muito grandes e levar muito tempo para serem carregados. Embora o Flash Player e o AIR permitam que o aplicativo reproduza sons mesmo antes de eles serem completamente carregados, talvez você queira dar ao usuário uma indicação da quantidade de dados de som carregada e da quantidade de som que já foi reproduzida.

A classe Sound despacha dois eventos que facilitam relativamente a exibição do progresso de carregamento de um som: `ProgressEvent.PROGRESS` e `Event.COMPLETE`. O exemplo a seguir mostra como usar esses eventos para exibir informações de progresso sobre o som que está sendo carregado:

```
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);
}
```

Esse código primeiro cria um objeto Sound e, em seguida, adiciona ouvintes para esse objeto para os eventos `ProgressEvent.PROGRESS` e `Event.COMPLETE`. Após o método `Sound.load()` ter sido chamado e os primeiros dados terem sido recebidos do arquivo de som, ocorre um evento `ProgressEvent.PROGRESS` que dispara o método `onSoundLoadProgress()`.

A porcentagem dos dados de som que foi carregada é igual ao valor da propriedade `bytesLoaded` do objeto gressEvent dividido pelo valor da propriedade `bytesTotal`. As mesmas propriedades `bytesLoaded` e `bytesTotal` estão disponíveis no objeto Sound também. O exemplo acima simplesmente mostra mensagens sobre o progresso do carregamento do som, mas é possível usar facilmente os valores de `bytesLoaded` e `bytesTotal` para atualizar os componentes da barra de progresso, como os provenientes da estrutura do Adobe Flex ou da ferramenta de autoria do Adobe Flash.

Esse exemplo também mostra como um aplicativo pode reconhecer e responder a um erro ao carregar arquivos de som. Por exemplo, se um arquivo de som com o nome fornecido não puder ser localizado, um evento `Event.IO_ERROR` será despachado pelo objeto Sound. No código anterior, o método `onIOError()` executa e exibe uma breve mensagem de erro quando ocorre um erro.

Trabalho com sons incorporados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O uso de sons incorporados, em vez do carregamento de som de um arquivo externo, é mais útil para pequenos sons que são usados como indicadores dentro da interface do usuário do aplicativo, como sons que são reproduzidos quando botões são clicados.

Quando um arquivo de som é incorporado no aplicativo, o tamanho do arquivo SWF resultante aumenta pelo tamanho do arquivo de som. Em outras palavras, a incorporação de arquivos de som grandes no aplicativo pode aumentar o tamanho do arquivo SWF para um tamanho indesejado.

O método exato a ser usado para incorporar um arquivo de som no arquivo SWF do aplicativo varia de acordo com o ambiente de desenvolvimento.

Uso de um arquivo de som incorporado no Flash

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A ferramenta de autoria do Flash permite importar sons de vários formatos e armazená-los como símbolos na Biblioteca. Em seguida, é possível atribuí-los a quadros na linha do tempo ou a quadros de um estado de botão, usá-los com Comportamentos ou usá-los diretamente no código do ActionScript. Esta seção descreve como usar sons incorporados no código ActionScript com a ferramenta de autoria do Flash. Para obter informações sobre as outras maneiras de usar sons incorporados no Flash, consulte “Importação de sons” em *Uso do Flash*.

Para incorporar um arquivo de som usando a Ferramenta de autoria do Flash:

1. Selecione Arquivo > Importar > Importar para biblioteca e, em seguida, selecione e importe um arquivo de som.
2. Clique com o botão direito do mouse no nome do arquivo importado no painel Biblioteca e selecione Propriedades. Clique na caixa de seleção Exportar para ActionScript.
3. No campo Classe, digite um nome a ser usado ao fazer referência a esse som incorporado no ActionScript. Por padrão, ele usará o nome do arquivo de som nesse campo. Se o nome do arquivo incluir um ponto, como no nome “DrumSound.mp3”, você deverá alterá-lo para algo como “DrumSound”. O ActionScript não permite um caractere ponto em um nome de classe. O campo Classe base ainda deve mostrar flash.media.Sound.

4 Clique em OK. Talvez você veja uma caixa de diálogo informando que uma definição dessa classe não pôde ser localizada no caminho de classe. Clique em OK e continue. Se você inseriu um nome de classe que não corresponde ao nome de qualquer uma das classes no caminho de classe do aplicativo, uma nova classe que herda da classe flash.media.Sound será gerada automaticamente para você.

5 Para usar o som incorporado, você faz referência ao nome da classe para aquele som no ActionScript. Por exemplo, o código a seguir começa criando uma nova ocorrência da classe DrumSound gerada automaticamente:

```
var drum:DrumSound = new DrumSound();
var channel:SoundChannel = drum.play();
```

DrumSound é uma subclasse da classe flash.media.Sound portanto ela herda os métodos e as propriedades da classe Sound, incluindo o método `play()` conforme mostrado acima.

Uso de um arquivo de som incorporado no Flex

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Existem várias maneiras de incorporar recursos de som em um aplicativo Flex, incluindo:

- Usar a tag de metadados `[Embed]` em um script
- Usar a diretiva `@Embed` no MXML para atribuir um recurso incorporado como uma propriedade de um componente, como Button ou SoundEffect.
- Usar a diretiva `@Embed` dentro de um arquivo CSS.

Esta seção descreve a primeira opção: como incorporar sons no código ActionScript dentro de um aplicativo Flex usando a tag de metadados `[Embed]`.

Para incorporar um ativo no código ActionScript, use a tag de metadados `[Embed]`.

Coloque o arquivo de som na pasta de origem principal ou em outra pasta que esteja no caminho de criação do projeto. Quando o compilador do encontra uma tag de metadados Embed, ele cria a classe de ativos incorporada para você. É possível acessar a classe por meio de uma variável de tipo de dados Class que pode ser declarada imediatamente após a tag de metadados `[Embed]`.

O código a seguir incorpora um som denominado smallSound.mp3 e usa uma variável denominada `soundClass` para armazenar uma referência à classe de ativos incorporada associada àquele som. Em seguida, o código cria uma ocorrência da classe de ativos incorporada, projeta-a como uma ocorrência da classe Sound e chama o método `play()` naquela ocorrência:

```
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();
        }
    }
}
```

Para usar o som incorporado para definir uma propriedade de um componente Flex, ele deve ser projetado como uma ocorrência da classe mx.core.SoundAsset em vez de uma ocorrência da classe Sound. Para obter um exemplo semelhante que usa a classe SoundAsset, consulte "Classes de ativos incorporadas" em Aprendendo o ActionScript 3.0.

Trabalho com arquivos de fluxo de som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A reprodução de um arquivo de som ou de vídeo enquanto seus dados ainda estão sendo carregados é conhecida como *streaming*. Arquivos de som externos que são carregados de um servidor remoto normalmente são transmitidos em fluxo para que o usuário não precise aguardar até que todos os dados do som sejam carregados para ouvir o som.

A propriedade `SoundMixer.bufferTime` representa o número de milissegundos de dados de som que o Flash Player ou o AIR deve coletar antes de permitir a reprodução do som. Em outras palavras, se a propriedade `bufferTime` estiver definida como 5000, o Flash Player ou o AIR carregarão pelo menos 5000 milissegundos de dados do arquivo de som antes de iniciar a reprodução do som. O valor padrão `SoundMixer.bufferTime` é 1000.

O aplicativo pode substituir o valor global `SoundMixer.bufferTime` de um som individual especificando explicitamente um novo valor de `bufferTime` ao carregar o som. Para substituir o tempo de buffer padrão, primeiro crie uma nova ocorrência da classe SoundLoaderContext, defina sua propriedade `bufferTime` e, em seguida, passe-a como um parâmetro para o método `Sound.load()`, conforme mostrado a seguir:

```
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();
```

Conforme a reprodução continua, o Flash Player e o AIR tentam manter o buffer de som do mesmo tamanho ou maior. Se os dados do som forem carregados mais rapidamente do que a velocidade de reprodução, a reprodução continuará sem interrupção. No entanto, se a taxa de carregamento de dados for reduzida devido a limitações da rede, o indicador de reprodução poderá atingir o final do buffer de som. Nesse caso, a reprodução será suspensa, embora ela seja reiniciada automaticamente quando mais dados de som forem carregados.

Para descobrir se a reprodução está suspensa porque o Flash Player ou o AIR estão aguardando o carregamento dos dados, use a propriedade `Sound.isBuffering`.

Trabalho com áudio gerado dinamicamente

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Nota: *A capacidade de gerar áudio dinamicamente está disponível a partir do Flash Player 10 e do Adobe AIR 1.5.*

Em vez de carregar ou transmitir em fluxo um som existente, é possível gerar dados de áudio dinamicamente. Você pode gerar dados de áudio quando estiver atribuindo um ouvinte de eventos ao evento `sampleData` de um objeto Sound. (O evento `sampleData` é definido na classe SampleDataEvent do pacote flash.events.) Neste ambiente, o objeto Sound não carrega dados de som de um arquivo. Em vez disso, ele funciona como um soquete para dados de som que estão sendo transmitidos em fluxo através do uso da função que você atribui a esse evento.

Quando você adiciona um ouvinte de eventos `sampleData` a um objeto Sound, o objeto periodicamente solicita dados para adicioná-los ao buffer de som. Esse buffer contém dados para o objeto Sound reproduzir. Quando você chama o método `play()` do objeto Sound, ele despacha o evento `sampleData` ao solicitar novos dados de som. (Isto só se aplica quando o objeto Sound não carregou dados mp3 de um arquivo.)

O objeto SampleDataEvent inclui uma propriedade `data`. No ouvinte de eventos, você grava objetos ByteArray neste objeto `data`. As matrizes de bytes gravadas neste objeto somam-se aos dados de som em buffer reproduzidos pelo objeto Sound. A matriz de bytes no buffer é um fluxo de valores de ponto flutuante que variam de -1 a 1. Cada valor de ponto flutuante representa a amplitude de um canal (esquerdo ou direito) de uma amostra de som. A amostragem do som ocorre a 44.100 amostras por segundo. Cada amostra contém um canal esquerdo e direito, intercalados como dados de ponto flutuante na matriz de bytes.

Na função do manipulador, use o método `ByteArray.writeFloat()` para gravar na propriedade `data` do evento `sampleData`. Por exemplo, o seguinte código gera uma onda senoidal:

```
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);
    }
}
```

Quando você chama `Sound.play()`, o aplicativo começa a chamar seu manipulador de eventos, solicitando dados de amostra de som. O aplicativo continua enviando eventos à medida que o som é reproduzido, até você parar de fornecer dados ou chamar `SoundChannel.stop()`.

A latência do evento varia de uma plataforma para outra e pode mudar em versões futuras do Flash Player e do AIR. Não dependa de uma latência específica; em vez disso, você deve calculá-la. Para fazer esse cálculo, use a seguinte fórmula:

```
(SampleDataEvent.position / 44.1) - SoundChannelObject.position
```

Forneça de 2048 a 8192 amostras para a propriedade `data` do objeto SampleDataEvent (para cada chamada ao ouvinte de eventos). Para obter o melhor desempenho, forneça o máximo possível de amostras (até 8192). Quanto menos amostras forem fornecidas, maior será a probabilidade de ocorrerem cliques e estalos durante a reprodução. Esse comportamento pode diferir em várias plataformas e pode ocorrer em diversas situações; por exemplo, no redimensionamento do navegador. O código que funciona em uma plataforma quando você fornece apenas 2048 amostras pode não funcionar tão bem quando executado em outra plataforma. Se precisar da menor latência possível, pense na possibilidade de tornar o volume de dados selecionável pelo usuário.

Se você fornecer menos de 2048 amostras (por chamada ao ouvinte de eventos `sampleData`), o aplicativo será interrompido depois de reproduzir as amostras restantes. Em seguida, o objeto SoundChannel despacha um evento SoundComplete.

Modificação do som de dados mp3

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Use o método `Sound.extract()` para extrair dados de um objeto Sound. Você pode usar (e modificar) esses dados para gravá-los no fluxo dinâmico de outro objeto Sound para fins de reprodução. Por exemplo, o código a seguir usa os bytes de um arquivo MP3 carregado e os passa usando uma função de filtro, `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;
}
```

Limitações quanto aos sons gerados

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

Quando você usa um ouvinte de eventos `sampleData` com um objeto Sound, os outros únicos métodos Sound ativados são `Sound.extract()` e `Sound.play()`. Chamar qualquer outro método ou propriedade resultará em uma exceção. Todos os métodos e propriedades do objeto SoundChannel continuam ativados.

Reprodução de sons

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A reprodução de um som carregado pode ser tão simples quanto chamar o método `Sound.play()` para um objeto Sound, da seguinte maneira:

```
var snd:Sound = new Sound(new URLRequest("smallSound.mp3"));
snd.play();
```

Ao reproduzir sons usando o ActionScript 3.0, é possível executar as seguintes operações:

- Reproduzir um som a partir de uma posição inicial específica.

- Pausar um som e reiniciar a reprodução a partir da mesma posição mais tarde.
- Saber exatamente quando a reprodução de um som é concluída.
- Rastrear o progresso da reprodução de um som.
- Alterar o volume ou a panorâmica enquanto o som é reproduzido.

Para executar essas operações durante a reprodução, use as classes SoundChannel, SoundMixer e SoundTransform.

A classe SoundChannel controle a reprodução de um único som. A propriedade `SoundChannel.position` pode ser considerada como um indicador de reprodução, indicando o ponto atual nos dados do som que está sendo reproduzido.

Quando um aplicativo chama o método `Sound.play()`, uma nova ocorrência da classe SoundChannel é criada para controlar a reprodução.

O aplicativo pode reproduzir um som a partir de uma posição inicial específica passando aquela posição, em termos de milissegundos, como o parâmetro `startTime` do método `Sound.play()`. Ele também pode especificar o número de vezes para repetição do som em sucessão rápida passando um valor numérico no parâmetro `loops` do método `Sound.play()`.

Quando o método `Sound.play()` é chamado com um parâmetro `startTime` e um parâmetro `loops`, o som é reproduzido repetidamente a partir do mesmo ponto inicial, conforme mostrado no código a seguir:

```
var snd:Sound = new Sound(new URLRequest("repeatingSound.mp3"));
snd.play(1000, 3);
```

Neste exemplo, o som é reproduzido a partir de um ponto um segundo após o início do som, três vezes em sucessão.

Pausa e reinício de um som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se o aplicativo reproduzir sons longos, como canções ou podcasts, talvez você queira permitir que os usuários pausem e reiniciem a reprodução desses sons. Um som não pode ser pausado literalmente durante a reprodução no ActionScript. Ele pode apenas ser interrompido. No entanto um som pode ser reproduzido a partir de um ponto qualquer. É possível registrar a posição do som na hora em que foi interrompido e, em seguida, reproduzir o som mais tarde a partir daquela posição.

Por exemplo, suponha que o código carrega e reproduz um arquivo de som como este:

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var channel:SoundChannel = snd.play();
```

Enquanto o som é reproduzido, a propriedade `SoundChannel.position` indica o ponto no arquivo de som que está em reprodução no momento. O aplicativo pode armazenar o valor da posição antes de interromper a reprodução do som, da seguinte maneira:

```
var pausePosition:int = channel.position;
channel.stop();
```

Para reiniciar a reprodução do som, passe o valor da posição armazenado anteriormente para reiniciar o som a partir do mesmo ponto em que foi interrompido anteriormente.

```
channel = snd.play(pausePosition);
```

Monitoramento da reprodução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo talvez queira saber quando a reprodução de um som é interrompida para que possa iniciar a reprodução de outro som ou limpar alguns recursos usados durante a reprodução anterior. A classe SoundChannel despacha um evento `Event.SOUND_COMPLETE` quando a reprodução do som é concluída. O aplicativo pode ouvir esse evento e tomar a ação apropriada, conforme mostrado a seguir:

```
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.");
}
```

A classe SoundChannel não despacha eventos de progresso durante a reprodução. Para relatar o progresso da reprodução, o aplicativo pode configurar seu próprio mecanismo de controle de tempo e rastrear a posição do indicador de reprodução do som.

Para calcular qual porcentagem de um som foi reproduzida, é possível dividir o valor da propriedade `SoundChannel.position` pelo comprimento dos dados do som que está sendo reproduzido:

```
var playbackPercent:uint = 100 * (channel.position / snd.length);
```

No entanto esse código relatará porcentagens exatas da reprodução apenas se os dados do som foram carregados completamente antes do início da reprodução. A propriedade `Sound.length` mostra o tamanho dos dados do som que estão sendo carregados no momento, não o tamanho eventual do arquivo de som inteiro. Para rastrear o progresso da reprodução de um fluxo de som que ainda está sendo carregado, o aplicativo deve estimar o tamanho eventual do arquivo de som inteiro e usar esse valor em seus cálculos. É possível estimar o comprimento eventual dos dados do som usando as propriedades `bytesLoaded` e `bytesTotal` do objeto Sound, da seguinte maneira:

```
var estimatedLength:int =
    Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
var playbackPercent:uint = 100 * (channel.position / estimatedLength);
```

O código a seguir carrega um arquivo de som maior e usa o evento `Event.ENTER_FRAME` como seu mecanismo de controle de tempo para mostrar o progresso da reprodução. Ele relata periodicamente a porcentagem de reprodução que é calculada como o valor da posição inicial dividido pelo comprimento total dos dados do som:

```
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);
}
```

Após o início do carregamento dos dados do som, esse código chama o método `snd.play()` e armazena o objeto SoundChannel resultante na variável `channel`. Em seguida, ele adiciona um ouvinte de eventos ao aplicativo principal para o evento `Event.ENTER_FRAME` e outro ouvinte de eventos ao objeto SoundChannel para o evento `Event.SOUND_COMPLETE` que ocorre quando a reprodução é concluída.

A cada vez que o aplicativo atinge um novo quadro em sua animação, o método `onEnterFrame()` é chamado. O método `onEnterFrame()` estima o comprimento total do arquivo de som com base na quantidade de dados que já foi carregada e, em seguida, calcula e exibe a porcentagem de reprodução atual.

Quando todo o som foi reproduzido, o método `onPlaybackComplete()` é executado, removendo o ouvinte do evento `Event.ENTER_FRAME` de forma que ele não tente exibir atualizações do progresso após a conclusão da reprodução.

O evento `Event.ENTER_FRAME` pode ser despachado muitas vezes por segundo. Em alguns casos, não é conveniente exibir o progresso da reprodução com tanta frequência. Nesses casos, o aplicativo pode configurar seu próprio mecanismo de controle do tempo usando a classe flash.util.Timer. Consulte "Trabalho com datas e horas" na página 1.

Interrupção de fluxos de som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Há uma peculiaridade no processo de reprodução de sons que estão sendo transmitidos em fluxo, isto é, sons que ainda estão sendo carregados enquanto estão sendo reproduzidos. Quando o aplicativo chama o método `SoundChannel.stop()` em uma ocorrência de SoundChannel que está reproduzindo um fluxo de som, a reprodução do som para em um quadro e, em seguida, no próximo quadro, ela reinicia a partir do início do som. Isso ocorre porque o processo de carregamento do som ainda está em execução. Para interromper o carregamento e a reprodução de um fluxo de som, chame a o método `Sound.close()`.

Considerações sobre segurança ao carregar e reproduzir sons

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A capacidade do aplicativo de acessar dados de som podem ser limitadas de acordo com o modelo de segurança do Flash Player ou do AIR. Cada som está sujeito a restrições de duas caixas de proteção de segurança, a caixa de proteção do próprio conteúdo (a "caixa de proteção do conteúdo") e a caixa de proteção do aplicativo ou do objeto que carrega e reproduz o som (a "caixa de proteção do proprietário"). Para conteúdo de aplicativo do AIR na caixa de proteção do aplicativo, todos os sons, inclusive aqueles carregados de outros domínios, são acessíveis ao conteúdo na caixa de proteção de segurança do aplicativo. No entanto conteúdo em outras caixas de proteção de segurança observam as mesmas regras que o conteúdo em execução no Flash Player. Para obter mais informações sobre o modelo de segurança do Flash Player em geral e sobre a definição de caixas de proteção, consulte "Segurança" na página 1079.

A caixa de proteção do conteúdo controla se dados de som detalhados podem ser extraídos do som usando a propriedade `id3` ou o método `SoundMixer.computeSpectrum()`. Ele não restringe o carregamento ou a reprodução do próprio arquivo de som.

O domínio de origem do arquivo de som define as limitações de segurança da caixa de proteção do conteúdo. Geralmente, se um arquivo de som está localizado no mesmo domínio ou pasta que o arquivo SWF do aplicativo ou do objeto que o carrega, o aplicativo ou o objeto terá acesso total àquele arquivo de som. Se o som for proveniente de um domínio diferente do domínio do aplicativo, ele ainda poderá ser trazido para a caixa de proteção do conteúdo usando um arquivo de política.

O aplicativo pode passar um objeto SoundLoaderContext com uma propriedade `checkPolicyFile` como parâmetro para o método `Sound.load()`. A configuração da propriedade `checkPolicyFile` como `true` instrui o Flash Player ou o AIR a procurar um arquivo de política no servidor onde o som é carregado. Se existir um arquivo de política e ele conceder acesso ao domínio do arquivo SWF que está sendo carregado, o arquivo SWF poderá carregar o arquivo de som, acessar a propriedade `id3` do objeto Sound e chamar o método `SoundMixer.computeSpectrum()` de sons carregados.

A caixa de proteção do proprietário controla a reprodução local dos sons. O aplicativo ou o objeto que inicia a reprodução de um som define a caixa de proteção do proprietário.

O método `SoundMixer.stopAll()` silencia os sons em todos os objetos SoundChannel que estão sendo reproduzidos no momento, desde que atendam aos seguintes critérios:

- Os sons foram iniciados por objetos dentro da mesma caixa de proteção do proprietário.
- Os sons são provenientes de uma origem com um arquivo de política que concede acesso ao domínio do aplicativo ou objeto que chama o método `SoundMixer.stopAll()`.

Contudo, em um aplicativo do AIR, o conteúdo na caixa de proteção de segurança do aplicativo (conteúdo instalado com o aplicativo do AIR) não é restringido por essas limitações de segurança.

Para descobrir se o método `SoundMixer.stopAll()` realmente interromperá todas os sons em reprodução, o aplicativo pode chamar o método `SoundMixer.areSoundsInaccessible()`. Se esse método retornar um valor de `true`, alguns dos sons que estão sendo reproduzidos estarão fora do controle da caixa de proteção do proprietário atual e não serão interrompidos pelo método `SoundMixer.stopAll()`.

O método `SoundMixer.stopAll()` também interrompe a continuação do indicador da reprodução para todos os sons que foram carregados de arquivos externos. No entanto sons que foram incorporados em arquivos FLA e anexados a quadros na linha do tempo usando a ferramenta de autoria do Flash talvez iniciem a reprodução novamente se a animação mover para um novo quadro.

Controle do volume e do panorama do som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um objeto SoundChannel individual controla os canais estéreos esquerdo e direito para um som. Se um som mp3 for um som monofônico, os canais estéreos esquerdo e direito do objeto SoundChannel conterão formas de onda idênticas.

É possível descobrir a amplitude de cada canal estéreo do som que está sendo reproduzido usando as propriedades `leftPeak` e `rightPeak` do objeto SoundChannel. Essas propriedades mostram a amplitude de pico da própria forma de onda do som. Elas não representam o volume de reprodução real. O volume de reprodução real é uma função da amplitude da onda do som e dos valores de volume definidos no objeto SoundChannel e na classe SoundMixer.

A propriedade pan de um objeto SoundChannel pode ser usada para especificar um nível diferente de volume para cada um dos canais esquerdo e direito durante a reprodução. A propriedade pan pode ter um valor variando de -1 a 1, em que -1 significa que o canal esquerdo reproduz no volume máximo enquanto o canal direito está silencioso e 1 significa que o canal direito reproduz no volume máximo enquanto o canal esquerdo está silencioso. Valores numéricos entre -1 e 1 definem valores proporcionais para os valores dos canais esquerdo e direito e um valor igual a 0 significa que os dois canais são reproduzidos em um nível equilibrado de nível de volume médio.

O código de exemplo a seguir cria um objeto SoundTransform com um valor de volume de 0,6 e um valor de pan de -1 (volume máximo no canal esquerdo e nenhum volume no canal direito). Ele passa o objeto SoundTransform como um parâmetro para o método `play()` que aplica esse objeto SoundTransform ao novo objeto SoundChannel que é criado para controlar a reprodução.

```
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);
```

É possível alterar o volume e a panorâmica enquanto um som está sendo reproduzido definindo as propriedades `pan` ou `volume` de um objeto SoundTransform e aplicando esse objeto como a propriedade `soundTransform` de um objeto SoundChannel.

Também é possível definir os valores globais de volume e de pan para todos os sons de uma vez usando a propriedade `soundTransform` da classe SoundMixer, conforme mostrado no exemplo a seguir:

```
SoundMixer.soundTransform = new SoundTransform(1, -1);
```

Também é possível usar um objeto SoundTransform para definir os valores de volume e panorâmica para um objeto Microphone (consulte "Captura de entrada do som" na página 474) e para objetos Sprite e SimpleButton.

O exemplo a seguir alterna a panorâmica do som do canal esquerdo para o canal direito e vice-versa enquanto o som é reproduzido.

```
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);
}
```

Esse código é iniciado carregando um arquivo de som e criando um novo objeto SoundTransform com o volume definido como 1 (volume total) e panorâmica definida como 0 (equilibrada igualmente entre a esquerda e a direita). Em seguida, ele chama o método `snd.play()`, passando o objeto SoundTransform como um parâmetro.

Enquanto o som é reproduzido, o método `onEnterFrame()` é executado repetidamente. O método `onEnterFrame()` usa a função `Math.sin()` para gerar um valor entre -1 e 1, uma faixa que corresponde aos valores aceitáveis da propriedade `SoundTransform.pan`. A propriedade `pan` do objeto SoundTransform é definida como o novo valor e, em seguida, a propriedade `soundTransform` do canal é definida para usar o objeto SoundTransform alterado.

Para executar esse exemplo, substitua o nome do arquivo bigSound.mp3 pelo nome de um arquivo mp3 local. Em seguida, execute o exemplo. Você deve ouvir o volume do canal esquerdo se tornando mais alto enquanto o volume do canal direito se torna mais suave e vice-versa.

Nesse exemplo, o mesmo efeito pode ser obtido definindo a propriedade `soundTransform` da classe SoundMixer. No entanto isso afetará a panorâmica de todos os sons que estão sendo reproduzidos no momento, não apenas o único som que está sendo reproduzido pelo objeto SoundChannel.

Trabalho com metadados de som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Arquivos de som que usam o formato mp3 podem conter dados adicionais sobre o som na forma de tags ID3.

Nem todo arquivo mp3 contém metadados ID3. Quando um objeto Sound carrega um arquivo de som mp3, ele despachará um evento `Event.ID3` se o arquivo de som contiver metadados ID3. Para evitar erros em tempo de execução, o aplicativo deve aguardar até receber o evento `Event.ID3` antes de acessar a propriedade `Sound.id3` de um som carregado.

O código a seguir mostra como reconhecer quando os metadados ID3 de um arquivo de som foram carregados:

```
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]);
    }
}
```

Esse código começa criando um objeto Sound e indicando que ele ouça o evento `Event.ID3`. Quando os metadados ID3 do arquivo de som são carregados, o método `onID3InfoReceived()` é chamado. O destino do objeto Event que é passado para o método `onID3InfoReceived()` é o objeto Sound original, portanto o método obtém a propriedade `id3` do objeto Sound e percorre todas as suas propriedades nomeadas para rastrear seus valores.

Acesso a dados de som brutos

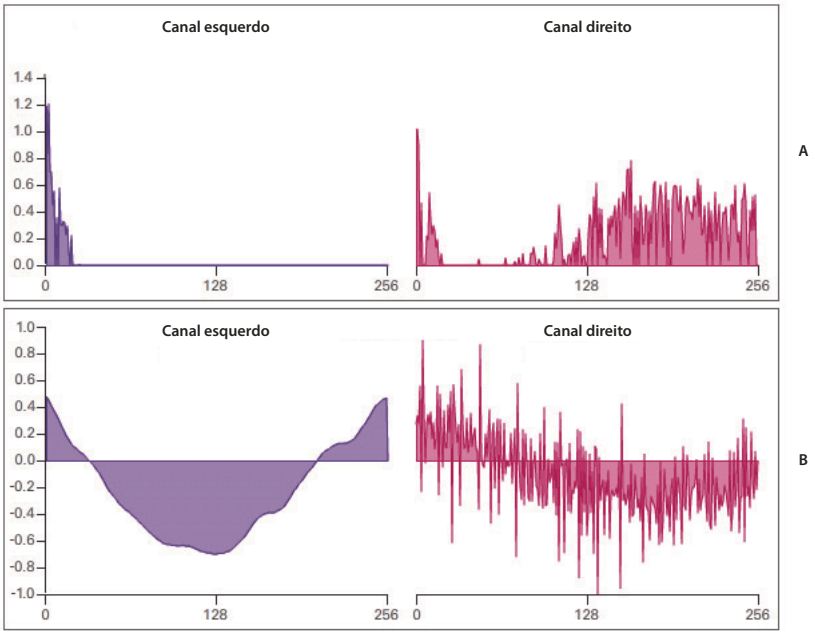
Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `SoundMixer.computeSpectrum()` permite que um aplicativo leia os dados de som brutos para a forma de onda que está sendo reproduzida no momento. Se mais de um objeto SoundChannel estiver sendo reproduzido, o método `SoundMixer.computeSpectrum()` mostrará os dados de som combinados de cada objeto SoundChannel misturado.

Os dados de som são retornados como um objeto ByteArray que contém 512 bytes de dados, cada um deles contendo um valor de ponto flutuante entre -1 e 1. Esses valores representam a amplitude dos pontos na forma de onda do som que está sendo reproduzido. Os valores são entregues em dois grupos de 256, o primeiro grupo do canal estéreo esquerdo e o segundo grupo do canal estéreo direito.

O método `SoundMixer.computeSpectrum()` retornará os dados do espectro da frequência em vez dos dados da forma de onda se o parâmetro `FFTMode` estiver definido como `true`. O espectro da frequência mostra a amplitude organizada pela frequência do som, da frequência mais baixa para a mais alta. Uma FFT (Transformação rápida de Fourier) é usada para converter os dados da forma de onda nos dados do espectro da frequência. Os valores do espectro da frequência resultantes variam de 0 a aproximadamente 1,414 (a raiz quadrada de 2).

O diagrama a seguir compara os dados retornados do método `computeSpectrum()` quando o parâmetro `FFTMode` está definido como `true` e quando está definido como `false`. O som cujos dados foram usados para esse diagrama contém um som grave alto no canal esquerdo e um som de toque de tambor no canal direito.



Valores retornados pelo método SoundMixer.computeSpectrum()
A. *fftMode=true* ***B.*** *fftMode=false*

O método `computeSpectrum()` também pode retornar dados que foram amostrados novamente em uma taxa de bits mais baixa. Geralmente, isso resulta em dados de forma de onda mais suaves ou em dados de frequência em detrimento de detalhes. O parâmetro `stretchFactor` controla a taxa na qual os dados do método `computeSpectrum()` são amostrados. Quando o parâmetro `stretchFactor` está definido como 0, o padrão, os dados do som são amostrados a uma taxa de 44,1 kHz. A taxa é dividida em dois em cada valor sucessivo do parâmetro stretchFactor, portanto um valor de 1 especifica uma taxa de 22,05 kHz, um valor de 2 especifica uma taxa de 11,025 kHz e assim por diante. O método `computeSpectrum()` ainda retorna 256 bytes por canal estéreo quando um valor de `stretchFactor` mais alto é usado.

O método `SoundMixer.computeSpectrum()` tem algumas limitações:

- Como os dados do som de um microfone ou de fluxos RTMP não passam pelo objeto global SoundMixer, o método `SoundMixer.computeSpectrum()` não retorna dados dessas origens.
- Se um ou mais dos sons que estão sendo reproduzidos de origens externas à caixa de proteção do conteúdo atual, restrições de segurança farão com que o método `SoundMixer.computeSpectrum()` emita um erro. Para obter mais detalhes sobre limitações de segurança do método `SoundMixer.computeSpectrum()` consulte "Considerações sobre segurança ao carregar e reproduzir sons" na página 467 e"Acesso à mídia carregada como dados" na página 1105.

Contudo, em um aplicativo do AIR, o conteúdo na caixa de proteção de segurança do aplicativo (conteúdo instalado com o aplicativo do AIR) não é restringido por essas limitações de segurança.

Criação de um único visualizador de som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

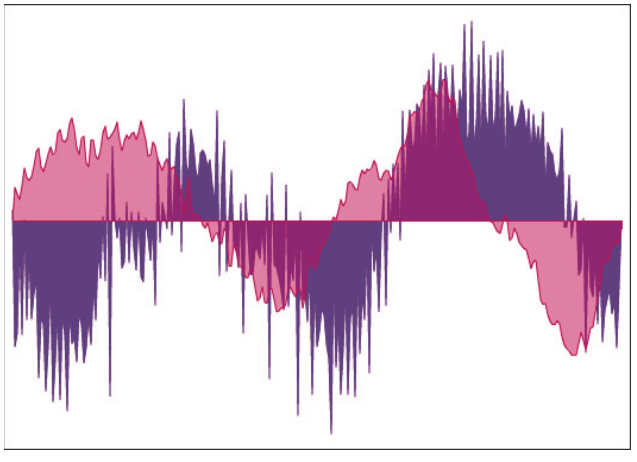
O exemplo a seguir usa o método `SoundMixer.computeSpectrum()` para mostrar um gráfico da forma de onda do som que é animado com cada quadro:

```
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);
}
```

Esse exemplo primeiro carrega e reproduz um arquivo de som e, em seguida, ouve o evento `Event.ENTER_FRAME` que disparará o método `onEnterFrame()` enquanto o som é reproduzido. O método `onEnterFrame()` é iniciado chamando o método `SoundMixer.computeSpectrum()` que armazena dos dados da onda do som no objeto ByteArray de `bytes`.

A forma de onda do som é plotada usando a API de desenho do vetor. Um loop `for` percorre os primeiros 256 valores de dados que representam o canal estéreo esquerdo e desenha uma linha de cada ponto até o próximo usando o método `Graphics.lineTo()`. Um segundo loop `for` percorre o próximo conjunto de 256 valores, plotando-os na ordem reversa desta vez, da direita para a esquerda. Os desenhos da forma de onda resultantes podem produzir um efeito interessante de imagem espelhada, conforme mostrado na imagem a seguir.



Captura de entrada do som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Microphone permite que o aplicativo se conecte a um microfone ou a outro dispositivo de entrada de som no sistema do usuário e transmita o áudio de entrada para os alto-falantes daquele sistema ou envie os dados de áudio a um servidor remoto, como o Flash Media Server. Você pode acessar os dados de áudio brutos a partir do microfone e gravá-los ou processá-los. Também pode enviar o áudio diretamente para os alto-falantes do sistema ou enviar dados de áudio compactados para um servidor remoto. É possível usar o codec Speex ou Nellymoser para os dados enviados a um servidor remoto. (O suporte para o codec Speex foi disponibilizado a partir do Flash Player 10 e do Adobe AIR 1.5.)

Mais tópicos da Ajuda

Michael Chaize: AIR, Android e o Microfone

Christophe Coenraets: notas de voz para o Android

Acesso a um microfone

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Microphone não tem um método construtor. Em vez disso, você usa o método estático `Microphone.getMicrophone()` para obter uma nova ocorrência de Microphone, conforme mostrado a seguir:

```
var mic:Microphone = Microphone.getMicrophone();
```

A chamada do método `Microphone.getMicrophone()` sem um parâmetro retorna o primeiro dispositivo de entrada de som descoberto no sistema do usuário.

Um sistema pode ter mais de um dispositivo de entrada de som conectado a ele. O aplicativo pode usar a propriedade `Microphone.names` para obter uma matriz dos nomes de todos os dispositivos de entrada de som disponíveis. Em seguida, ele pode chamar o método `Microphone.getMicrophone()` com um parâmetro `index` que corresponde ao valor de índice de um nome de dispositivo na matriz.

Um sistema talvez não tenha um microfone ou outro dispositivo de entrada de som conectado a ele. É possível usar a propriedade `Microphone.names` ou o método `Microphone.getMicrophone()` para verificar se o usuário tem um dispositivo de entrada de som instalado. Se o usuário não tiver um dispositivo de entrada de som instalado, a matriz `names` terá um comprimento de zero e o método `getMicrophone()` retornará um valor `null`.

Quando o aplicativo chama o método `Microphone.getMicrophone()`, o Flash Player exibe a caixa de diálogo Configurações do Flash Player que solicita que o usuário permita ou negue acesso do Flash Player à câmara e ao microfone no sistema. Depois que o usuário clicar no botão Permitir ou Negar neste diálogo, um StatusEvent será despachado. A propriedade `code` dessa ocorrência de StatusEvent indica se o acesso ao microfone foi permitido ou negado, conforme mostrado neste exemplo:

```
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.");
    }
}
```

A propriedade `StatusEvent.code` conterá “Microphone.Unmuted”, se o acesso for permitido, ou “Microphone.Muted”, se o acesso for negado.

A propriedade `Microphone.muted` é definida como `true` ou `false` quando o usuário permite ou nega acesso ao microfone, respectivamente. No entanto, a propriedade `muted` não é definida na ocorrência de Microphone até que StatusEvent tenha sido despachado, de modo que seu aplicativo também deve esperar até o evento `StatusEvent.STATUS` ser despachado antes de verificar a propriedade `Microphone.muted`.

Para que o Flash Player exiba a caixa de diálogo de configurações, a janela de aplicativo precisa ser grande o suficiente para exibi-la (pelo menos 215 x 138 pixels). Caso contrário, o acesso é negado automaticamente.

O conteúdo em execução na caixa de proteção do aplicativo do AIR não precisa da permissão do usuário para acessar o microfone. Assim, eventos de status para ligar e desligar o som do microfone nunca são despachados. O conteúdo em execução no AIR fora da caixa de proteção do aplicativo requer a permissão do usuário, de modo que esses eventos de status podem ser despachados.

Roteamento de áudio do microfone para alto-falantes locais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A entrada de áudio de um microfone pode ser roteada para os alto-falantes do sistema local chamando o método `Microphone.setLoopback()` com um valor de parâmetro de `true`.

Quando o som de um microfone local é roteado para alto-falantes locais, há um risco de criar um loop de feedback de áudio, o que pode provocar sons estridentes e danificar o hardware de som. A chamada do método `Microphone.setUseEchoSuppression()` com um valor de parâmetro `true` reduz, mas não elimina completamente, o risco de ocorrência de feedback de áudio. A Adobe recomenda sempre chamar `Microphone.setUseEchoSuppression(true)` antes de chamar `Microphone.setLoopback(true)`, a menos que você tenha certeza de que o usuário esteja reproduzindo o som usando fones de ouvido ou outra coisa que não seja um alto-falante.

O código a seguir mostra como rotear o áudio de um microfone local para os alto-falantes do sistema local:

```
var mic:Microphone = Microphone.getMicrophone();
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
```

Alteração do áudio do microfone

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo pode alterar os dados de áudio de um microfone de duas maneiras. Primeiro, ele pode alterar o ganho do som de entrada, o que efetivamente multiplica os valores de entrada por um valor especificado para criar um som mais baixo ou mais silencioso. A propriedade `Microphone.gain` aceita valores numéricos entre 0 e 100 inclusive. Um valor de 50 funciona como um multiplicador de um e especifica o volume normal. Um valor de zero funciona como um multiplicador de zero e silencia efetivamente o áudio de entrada. Valores acima de 50 especificam um volume mais alto do que o normal.

O aplicativo também pode alterar a taxa de amostragem do áudio de entrada. Taxas de amostragem mais altas aumentam a qualidade do som, mas elas também criam fluxos de dados mais densos que usam mais recursos para transmissão e armazenamento. A propriedade `Microphone.rate` representa a taxa de amostragem de áudio medida em quilohertz (kHz). A taxa de amostragem padrão é de 8 kHz. É possível definir a propriedade `Microphone.rate` como um valor mais alto do que 8 kHz se o microfone oferecer suporte à taxa mais alta. Por exemplo, configurar a propriedade `Microphone.rate` com um valor de 11 define a taxa de amostragem como 11 kHz. Configurá-la como 22 define a taxa de amostragem como 22 kHz e assim por diante. As taxas de amostra disponíveis dependem do codec selecionado. Quando você usa o codec Nellymoser, pode especificar 5, 8, 11, 16, 22 e 44 kHz como taxa de amostra. Quando você usa o codec Speex (disponível a partir do Flash Player 10 e do Adobe AIR 1.5), só pode usar 16 kHz.

Detecção de atividade do microfone

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para conservar a largura de banda e os recursos de processamento, o Flash Player tenta detectar quando nenhum som está sendo transmitido por um microfone. Quando o nível de atividade do microfone permanece abaixo do limite do nível de silêncio por um certo período, o Flash Player para de transmitir a entrada de áudio e despacha um simples ActivityEvent. Se você usar o codec Speex (disponível no Flash Player 10 ou em versões posteriores e no Adobe AIR 1.5 ou em versões posteriores), defina o nível de silêncio como 0 para assegurar que o aplicativo transmitirá dados de áudio continuamente. A detecção da atividade de voz do Speex automaticamente reduz a largura de banda.

Nota: *Um objeto Microphone despacha eventos Activity somente quando o aplicativo estiver monitorando o microfone. Assim, se você não chamar `setLoopBack( true )`, adicione um ouvinte para eventos de amostra de dados, ou anexe o microfone a um objeto NetStream e, em seguida, nenhum evento activity será despachado.*

Três propriedades da classe Microphone monitoram e controlam a detecção de atividade:

- A propriedade somente leitura `activityLevel` indica a quantidade de som que o microfone está detectando em uma escala de 0 a 100.
- A propriedade `silenceLevel` especifica a quantidade de som necessária para ativar o microfone e despachar um evento `ActivityEvent.ACTIVITY`. A propriedade `silenceLevel` também usa uma escala de 0 a 100, e o valor padrão é 10.
- A propriedade `silenceTimeout` descreve o número de milissegundos que o nível de atividade deve permanecer abaixo do nível de silêncio, até que um evento `ActivityEvent.ACTIVITY` seja despachado para indicar que o microfone está silencioso. O valor padrão de `silenceTimeout` é 2000.

As propriedades `Microphone.silenceLevel` e `Microphone.silenceTimeout` são somente leitura, mas seus valores podem ser alterados usando o método `Microphone.setSilenceLevel()`.

Em alguns casos, o processo de ativação do microfone quando nova atividade é detectada pode provocar um pequeno atraso. Manter o microfone ativo durante todo o tempo pode remover esses atrasos na ativação. O aplicativo pode chamar o método `Microphone.setSilenceLevel()` com o parâmetro `silenceLevel` definido como zero para indicar ao Flash Player para manter o microfone ativo e continuar a coletar dados de áudio, mesmo quando nenhum som está sendo detectado. De modo oposto, a configuração do parâmetro `silenceLevel` como 100 impede que o microfone seja ativado de qualquer modo.

O exemplo a seguir exibe informações sobre o microfone e relata eventos de atividade e eventos de status despachados por um objeto Microphone:

```
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);
}
```

Ao executar o exemplo acima, fale ou faça ruídos no microfone do sistema e observe as instruções de rastreamento resultantes serem exibidas em um console ou janela de depuração.

Envio e recebimento de áudio de um servidor de mídia

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Recursos adicionais de áudio também estão disponíveis ao usar o ActionScript com um servidor de mídia de fluxo contínuo, como o Flash Media Server.

Em particular, seu aplicativo pode conectar um objeto Microphone a um objeto NetStream e transmitir dados diretamente do microfone do usuário para o servidor. Os dados de áudio também podem ser transmitidos do servidor para um aplicativo e reproduzidos como parte de um MovieClip ou usando um objeto Video.

O suporte para o codec Speex foi disponibilizado a partir do Flash Player 10 e do Adobe AIR 1.5. Para definir o codec usado para áudio compactado enviado ao servidor de mídia, defina a propriedade `codec` do objeto Microphone. Esta propriedade pode ter dois valores, que são enumerados na classe SoundCodec. Definir a propriedade do codec como `SoundCodec.SPEEX` seleciona o codec Speex para compactação de áudio. Definir a propriedade como `SoundCodec.NELLYMOSER` (o padrão) seleciona o codec Nellymoser para compactação de áudio.

Para obter mais informações, consulte a documentação on-line do Flash Media Server em www.adobe.com/go/learn_fms_docs_br.

Captura dos dados de som do microfone

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

No Flash Player 10.1 e no AIR 2 ou posteriores, você pode capturar os dados de um microfone, como uma matriz de bytes dos valores de ponto flutuantes. Cada valor representa uma amostragem dos dados de áudio monofônicos.

Para obter dados de microfone, defina um ouvinte de evento para o evento `sampleData` do objeto Microphone. O objeto Microphone despacha os eventos `sampleData` periodicamente à medida que o buffer do microfone é preenchido com amostragens de som. O objeto SampleDataEvent tem uma propriedade `data` que é uma matriz de bytes das amostragens de som. As amostras são representadas como valores de ponto flutuantes, cada uma representando uma amostragem de som monofônico.

O código a seguir captura os dados de som do microfone em um objeto ByteArray chamado `soundBytes`:

```
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);
    }
}
```

Você pode reutilizar os bytes de amostra como áudio de reprodução para um objeto Sound. Se fizer isso, defina a propriedade `rate` do objeto Microphone como 44, que é a taxa de amostragem usada pelos objetos Sound. (Você também pode converter as amostragens de microfone capturadas a uma taxa inferior a 44 kHz que é exigida pelo objeto Sound.) Além disso, tenha em mente que o objeto Microphone captura as as amostragens monofônicas, ao passo que o objeto Sound usa som estéreo. Por isso, escreva cada um dos bytes capturados pelo objeto Microphone no objeto Sound duas vezes. O exemplo a seguir captura quatro segundos de dados do microfone e o reproduz novamente usando um objeto Sound:

```
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.");
}
```

Para obter mais informações sobre os sons de reprodução dos dados de amostragem do som, consulte “Trabalho com áudio gerado dinamicamente” na página 461.

Exemplo de som: Podcast Player

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um podcast é um arquivo de som distribuído pela Internet sob demanda ou por assinatura. Podcasts normalmente são publicados como parte de uma série, o que também é chamado de canal de podcast. Como episódios de podcast podem durar de um minuto a muitas horas, eles normalmente são transmitidos enquanto estão sendo carregados. Episódios de podcast, que também são chamados de itens, normalmente são entregues no formato de arquivo mp3. Os podcasts de vídeo também são populares, mas esse aplicativo de amostra reproduz apenas podcasts de áudio que usam arquivos mp3.

Este exemplo não é de um aplicativo agregador de podcast completo. Por exemplo, ele não gerencia assinaturas para podcasts específicos ou lembra quais podcasts foram ouvidos pelo usuário na próxima vez que o aplicativo é executado. Ele pode funcionar como um ponto de partida para um agregador de podcast mais completo.

Este exemplo de Podcast Player ilustra as seguintes técnicas de programação do ActionScript:

- Leitura de um RSS feed externo e análise de seu conteúdo XML
- Criação de uma classe SoundFacade para simplificar o carregamento e a reprodução de arquivos de som.
- Exibição do progresso da reprodução do som.
- Pausa e reinício da reprodução do som.

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Podcast Player podem ser encontrados na pasta Amostras/PodcastPlayer. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
PodcastPlayer.mxml ou PodcastPlayer.fla	A interface do usuário do aplicativo para Flex (MXML) ou Flash (FLA).
comp/example/programmingas3/podcastplayer/PodcastPlayer.as	Classe Document que contém a lógica da interface de usuário do player de podcast (somente no Flash).
SoundPlayer.mxml	Um componente MXML que exibe botões de reprodução e barras de progresso e controla a reprodução de som, apenas para Flex.
main.css	Estilos da interface de usuário do aplicativo (somente no Flex).
images/	Ícones para aplicar estilo aos botões (somente no Flex).
comp/example/programmingas3/podcastplayer/SoundPlayer.as	Classe do símbolo de clipe de filme SoundPlayer que contém a lógica da interface de usuário do player de som (somente no Flash).
comp/example/programmingas3/podcastplayer/PlayButtonRenderer.as	Renderizador de células personalizado para exibir um botão Reproduzir em uma célula de grade de dados (somente no Flash).
com/example/programmingas3/podcastplayer/RSSBase.as	Uma classe base que fornece propriedades e métodos comuns para as classes RSSChannel e RSSItem.

Arquivo	Descrição
com/example/programmingas3/podcastplayer/RSSChannel.as	Uma classe ActionScript que mantém dados sobre um canal RSS.
com/example/programmingas3/podcastplayer/RSSItem.as	Uma classe ActionScript que mantém dados sobre um item RSS.
com/example/programmingas3/podcastplayer/SoundFacade.as	A classe principal do ActionScript para o aplicativo. Ela encapsula os métodos e eventos da classe Sound e da classe SoundChannel e adiciona suporte para pausar e reiniciar a reprodução.
com/example/programmingas3/podcastplayer/URLService.as	Uma classe ActionScript que recupera dados de uma URL remota.
playerconfig.xml	Um arquivo XML que contém uma lista de RSS feeds que representam canais de podcast.
comp/example/programmingas3/utils/DateUtil.as	Classe usada para facilitar a formatação de dados (somente no Flash).

Leitura de dados RSS para um canal de podcast

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo Podcast Player começa lendo informações sobre vários canais de podcast e seus episódios:

1. Primeiro, o aplicativo lê um arquivo de configuração XML que contém uma lista de canais de podcast e exibe a lista de canais para o usuário.

2. Quando o usuário seleciona um dos canais de podcast, ele lê o RSS feed do canal e exibe uma lista de episódios de canal.

Este exemplo usa a classe do utilitário URLLoader para recuperar dados com base em texto de um local remoto ou de um arquivo local. O Podcast Player primeiro cria um objeto URLLoader para obter uma lista de RSS feeds em formato XML do arquivo playerconfig.xml. Em seguida, quando o usuário seleciona um feed específico da lista, um novo objeto URLLoader é criado para ler os dados RSS daquela URL do feed.

Simplificação do carregamento e da reprodução de som usando a classe SoundFacade

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A arquitetura de som do ActionScript 3.0 é poderosa, mas complexa. Aplicativos que precisam apenas de recursos básicos de carregamento e reprodução de som podem usar uma classe que oculte um pouco da complexidade fornecendo um conjunto mais simples de chamadas e eventos de métodos. No mundo de padrões de design de software, essa classe é chamada de *facade*.

A classe SoundFacade apresenta uma única interface para executar as seguintes tarefas:

- Carregamento de arquivos de som usando um objeto Sound, um objeto SoundLoaderContext e uma classe SoundMixer.
- Reprodução de arquivos de som usando os objetos Sound e SoundChannel.
- Despacho de eventos de progresso da reprodução.

- Pausa e reinício da reprodução do som usando os objetos Sound e SoundChannel.

A classe SoundFacade tenta oferecer mais da funcionalidade das classes de som do ActionScript com menos complexidade.

O código a seguir mostra a declaração da classe, as propriedades da classe e o método construtor `SoundFacade()`:

```
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();
        }
    }
```

A classe SoundFacade estende a classe EventDispatcher para poder despachar seus próprios eventos. O código da classe primeiro declara propriedades de um objeto Sound e de um objeto SoundChannel. A classe também armazena o valor da URL do arquivo de som e uma propriedade `bufferTime` a ser usada ao transmitir o som em fluxo. Além disso, ele aceita alguns valores de parâmetros boolianos que afetam o comportamento do carregamento e da reprodução:

- O parâmetro `autoLoad` indica ao objeto que o carregamento do som deve iniciar assim que esse objeto é criado.
- O parâmetro `autoPlay` indica que a reprodução do som deve iniciar assim que dados de som suficientes estiverem carregados. Se esse for um fluxo de som, a reprodução será iniciada assim que dados suficientes, conforme especificado pela propriedade `bufferTime`, estiverem carregados.
- O parâmetro `streaming` indica que a reprodução desse arquivo pode ser iniciada antes do carregamento ser concluído.

O parâmetro `bufferTime` é padronizado como um valor de -1. Se o método construtor detectar um valor negativo no parâmetro `bufferTime`, ele definirá a propriedade `bufferTime` como o valor de `SoundMixer.bufferTime`. Isso permite que o aplicativo seja padronizado para o valor global `SoundMixer.bufferTime` conforme desejado.

Se o parâmetro `autoLoad` for definido como `true`, o método construtor chamará imediatamente o método `load()` seguinte para iniciar o carregamento do arquivo de som:

```
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);
}
```

O método `load()` cria um novo objeto Sound e adiciona ouvintes para todos os eventos de som importantes. Em seguida, ele indica ao objeto Sound para carregar o arquivo de som usando um objeto SoundLoaderContext para passar o valor de `bufferTime`.

Como a propriedade `url` pode ser alterada, uma ocorrência de SoundFacade pode ser usada para reproduzir arquivos de som diferentes em sucessão: simplesmente altere a propriedade `url` e chame o método `load()` e o novo arquivo de som será carregado.

Os três métodos ouvintes de eventos a seguir mostram como o objeto SoundFacade rastreia o progresso do carregamento e decide quando iniciar a reprodução do som:

```
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();
    }
}
```

O método `onLoadOpen()` é executado quando o carregamento do som é iniciado. Se o som puder ser reproduzido em modo de streaming, o método `onLoadComplete()` definirá o sinalizador `isReadyToPlay` como `true` imediatamente. O sinalizador `isReadyToPlay` determina se o aplicativo pode iniciar a reprodução do som, talvez em resposta a uma ação do usuário, como um clique em um botão de reprodução. A classe SoundChannel gerencia o buffer de dados de som, portanto não há necessidade de verificar explicitamente se dados suficientes foram carregados antes de chamar o método `play()`.

O método `onLoadProgress()` é executado periodicamente durante o processo de carregamento. Ele simplesmente despacha um clone de seu objeto ProgressEvent para uso pelo código que usa esse objeto SoundFacade.

Quando os dados do som estiverem totalmente carregados, o método `onLoadComplete()` é executado, chamando o método `play()` para sons que não sejam de fluxo, se necessário. O próprio método `play()` é mostrado a seguir.

```
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();
        }
    }
}
```

O método `play()` chamará o método `Sound.play()` se o som estiver pronto para execução. O objeto SoundChannel resultante é armazenado na propriedade `sc`. Em seguida, o método `play()` cria um objeto Timer que será utilizado para despachar eventos de progresso da reprodução em intervalos regulares.

Exibição do progresso da reprodução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A criação de um objeto Timer para acionar o monitoramento da reprodução é uma operação complexa que você deve codificar apenas uma vez. O encapsulamento dessa lógica de Timer em uma classe reutilizável, como a classe SoundFacade, permite que os aplicativos ouçam alguns tipos de eventos de progresso quando um som está sendo carregado e quando ele está sendo reproduzido.

O objeto Timer criado pelo método `SoundFacade.play()` despacha uma ocorrência de TimerEvent a cada segundo. O método `onPlayTimer()` a seguir é executado sempre que um novo TimerEvent é recebido:

```
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);
}
```

O método `onPlayTimer()` implementa a técnica de estimativa de tamanho descrita na seção "Monitoramento da reprodução" na página 465. Em seguida, ele cria uma nova ocorrência de ProgressEvent com um tipo de evento de `SoundFacade.PLAY_PROGRESS`, com a propriedade `bytesLoaded` definida como a posição atual do objeto SoundChannel e a propriedade `bytesTotal` definida como o comprimento estimado dos dados do som.

Pausa e reinício da reprodução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `SoundFacade.play()` mostrado anteriormente aceita um parâmetro `pos` correspondente a uma posição inicial nos dados de som. Se o valor `pos` for zero, a reprodução do som será executada do início.

O método `SoundFacade.stop()` também aceita um parâmetro `pos` conforme mostrado aqui:

```
public function stop(pos:int = 0):void
{
    if (this.isPlaying)
    {
        this.pausePosition = pos;
        this.sc.stop();
        this.playTimer.stop();
        this.isPlaying = false;
    }
}
```

Sempre que o método `SoundFacade.stop()` é chamado, ele define a propriedade `pausePosition` de forma que o aplicativo saiba onde posicionar o indicador de reprodução se o usuário desejar retomar a reprodução do mesmo som.

Os métodos `SoundFacade.pause()` e `SoundFacade.resume()` mostrados a seguir chamam os métodos `SoundFacade.stop()` e `SoundFacade.play()` respectivamente, passando um valor de parâmetro `pos` a cada vez.

```
public function pause():void
{
    stop(this.sc.position);
}

public function resume():void
{
    play(this.pausePosition);
}
```

O método `pause()` passa o valor atual de `SoundChannel.position` para o método `play()` que armazena o valor na propriedade `pausePosition`. O método `resume()` inicia a reprodução do mesmo som novamente usando o valor de `pausePosition` como o ponto de início.

Extensão do exemplo do Podcast Player

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Este exemplo apresenta um Podcast Player reduzido ao essencial que demonstra o uso da classe SoundFacade reutilizável. É possível adicionar outros recursos para aprimorar a utilidade deste aplicativo, incluindo o seguinte:

- Armazenar a lista de feeds e informações de uso sobre cada episódio em uma ocorrência de SharedObject que pode ser usada na próxima vez que o usuário executar o aplicativo.
- Permitir que o usuário adicione seus próprios RSS feeds à lista de canais de podcast.
- Memorizar a posição do indicador de reprodução quando o usuário interromper ou sair de um episódio, para que ele possa ser reiniciado daquele ponto na próxima vez que o usuário executar o aplicativo.
- Baixar arquivos mp3 de episódios para ouvir offline, quando o usuário não estiver conectado à Internet.
- Adicionar recursos de assinatura que verificam periodicamente novos episódios em um canal de podcast e atualizar a lista de episódios automaticamente.
- Adicionar funcionalidade de pesquisa e de procura de podcast usando uma API de um serviço de host de podcast, como o Odeo.com.

Capítulo 25: Trabalho com vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os vídeos criados com o Adobe Flash são uma das tecnologias de destaque na Internet. No entanto, a apresentação tradicional do vídeo, em uma tela retangular com uma barra de progresso e botões de controle na parte inferior, é apenas um dos usos possíveis do vídeo. Por meio do ActionScript, você tem acesso e controle ajustados sobre o carregamento, a apresentação e a reprodução de vídeos.

Noções básicas sobre vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um recurso importante do Adobe® Flash® Player e do Adobe® AIR™ é a capacidade de exibir e manipular informações de vídeo com o ActionScript, da mesma forma que você pode manipular outros conteúdos visuais, como imagens, animação, texto e assim por diante. Ao criar um arquivo FLV (Flash Video) no Adobe Flash CS4 Professional, você tem a opção de selecionar uma capa que inclui controles de reprodução comuns. Todavia, não há motivo para se limitar às opções disponíveis. Usando o ActionScript, você tem controle ajustado sobre o carregamento, a exibição e a reprodução de vídeo, o que significa que é possível criar sua própria capa de player de vídeo ou usar o vídeo de qualquer maneira menos tradicional que você queira. O trabalho com vídeo no ActionScript envolve o trabalho com uma combinação de várias classes:

- Classe Video: a caixa clássica de conteúdo de vídeo no Palco é uma ocorrência da classe Video. A classe Video é um objeto de exibição, por isso pode ser manipulada com as mesmas técnicas aplicáveis a outros objetos de exibição, como posicionamento, aplicação de transformações, aplicação de filtros e modos de mesclagem, entre outras.
- Classe StageVideo: normalmente, a classe Video usa decodificação e renderização por software. Quando a aceleração por hardware da GPU está disponível em um dispositivo, seu aplicativo pode aproveitar melhor a apresentação acelerada por hardware alternando para a classe StageVideo. A API StageVideo inclui um conjunto de eventos que indica ao código para quando alternar entre os objetos StageVideo e Video. O vídeo Stage impõe algumas restrições de menor importância sobre a reprodução de vídeo. Se seu aplicativo aceita essas limitações, implemente a API do StageVideo. Consulte “Orientações e limitações” na página 530.
- Classe NetStream: quando você está carregando um arquivo de vídeo a ser controlado pelo ActionScript, uma instância de NetStream representa a origem do conteúdo do vídeo — neste caso, um fluxo de dados de vídeo. O uso de uma instância de NetStream também inclui o uso de um objeto NetConnection, que é a conexão com o arquivo de vídeo — como o túnel pelo qual é carregado o conteúdo do vídeo.
- Classe Camera: quando você está trabalhando com dados de vídeo de uma câmera conectada ao computador do usuário, uma instância de Camera representa a origem do conteúdo do vídeo — a câmera do usuário e os dados de vídeo disponibilizados por ela. Novidade do Flash Player 11.4 e AIR 3.4: é possível usar uma câmera como feed do StageVideo.

Quando você carrega vídeo externo, pode carregar o arquivo de um servidor Web padrão para download progressivo ou trabalhar com vídeo de fluxo contínuo entregue por um servidor especializado, como o Flash® Media Server da Adobe.

Conceitos e termos importantes

Ponto de sinalização Um marcador que pode ser colocado em um momento específico de um arquivo de vídeo, por exemplo, para funcionar como um marcador que localiza esse ponto no tempo ou para fornecer dados adicionais associados ao momento em questão.

Codificação O processo de pegar dados de vídeo em um formato e convertê-los em outro formato de dados de vídeo; por exemplo, você pode pegar um vídeo de uma origem de alta resolução e convertê-lo em um formato adequado para disponibilização na Internet.

Quadro Um único segmento de informações de vídeo; cada quadro é como uma imagem estática que representa um instantâneo de um momento. A execução de quadros em sequência em alta velocidade cria a ilusão de movimento.

Quadro principal Um quadro de vídeo que contém todas as informações do quadro. Outros quadros que vêm após um quadro-chave contêm apenas informações sobre no quê eles são diferentes do quadro-chave, e não todas as informações do quadro inteiro.

Metadados Informações sobre um arquivo de vídeo que são incorporadas a ele e recuperadas quando o vídeo é carregado.

Download progressivo Quando um arquivo de vídeo é disponibilizado de um servidor Web padrão, os dados de vídeo são carregados por meio de download progressivo, o que significa que as informações sobre o vídeo são carregadas em sequência. Isso tem a vantagem de que a reprodução do vídeo pode começar antes de terminar o download do arquivo inteiro; no entanto, o download progressivo impede de você avançar para uma parte do vídeo que ainda não foi carregada.

Streaming Como alternativa ao download progressivo, pode-se usar um servidor de vídeo especial para disponibilizar vídeo pela Internet usando uma técnica conhecida como streaming (também chamada de “true streaming”). Com o streaming, o computador do visualizador nunca baixa todo o vídeo de uma só vez. Para agilizar os tempos de download, em qualquer momento o computador só precisa de uma parte de todas as informações do vídeo. Como um servidor especial controla a entrega do conteúdo do vídeo, qualquer parte do vídeo pode ser acessada a qualquer momento, e não é necessário aguardar o término do download para acessá-lo.

Noções básicas sobre formatos de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Além do formato de vídeo Adobe FLV, o Flash Player e o Adobe AIR dão suporte a vídeo e áudio codificados em H.264 e HE-AAC a partir de formatos de arquivo com o padrão MPEG-4. Esses formatos transmitem vídeos de alta qualidade com taxas de bits mais baixas. Os desenvolvedores podem utilizar ferramentas padrão do setor, como o Adobe Premiere Pro e o Adobe After Effects, para criar e disponibilizar conteúdos em vídeo interessantes.

Tipo	Formato	Recipiente
Vídeo	H.264	MPEG-4: MP4, M4V, F4V, 3GPP
Vídeo	Sorenson Spark	Arquivo FLV
Vídeo	ON2 VP6	Arquivo FLV
Áudio	AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4:MP4, M4V, F4V, 3GPP

Tipo	Formato	Recipiente
Áudio	Mp3	Mp3
Áudio	Nellymoser	Arquivo FLV
Áudio	Speex	Arquivo FLV

Mais tópicos da Ajuda

Flash Media Server: codecs com suporte

Adobe HTTP Dynamic Streaming

Codificação de vídeo para dispositivos móveis

O AIR no Android pode decodificar uma grande variedade de vídeos H.264. Entretanto, somente um pequeno subconjunto de vídeos H.264 é adequado para reprodução suave em telefones celulares. Isso acontece porque muitos celulares são restringidos em termos de poder de processamento. O Adobe Flash Player para celulares pode decodificar vídeos H.264 usando a aceleração de hardware incorporada. Essa decodificação assegura melhor qualidade com consumo mais baixo de energia.

O padrão H.264 oferece suporte a várias técnicas de codificação. Somente dispositivos sofisticados reproduzem vídeo suavemente com perfis e níveis complexos. No entanto, a maioria dos dispositivos pode reproduzir vídeo codificado no perfil de linha e base. Em dispositivos móveis, a aceleração por hardware está disponível para um subconjunto dessas técnicas. Os parâmetros de perfil e de nível definem esse subconjunto de técnicas de codificação e as configurações usadas pelo codificador. Para os desenvolvedores, isso significa codificar o vídeo na resolução selecionada, que é reproduzida satisfatoriamente na maioria dos dispositivos.

Embora as resoluções que se beneficiam da aceleração por hardware variem de acordo com o dispositivo, a maioria dos dispositivos oferece suporte às resoluções padrão a seguir.

Proporções	Resoluções recomendadas		
4:3	640 × 480	512 × 384	480 × 360
16:9	640 × 360	512 x 288	480 × 272

Nota: *O Flash Player oferece suporte a cada nível e perfil do padrão H.264. Acatar estas recomendações garante a aceleração por hardware e uma melhor experiência do usuário na maioria dos dispositivos. Estas recomendações não são obrigatórias.*

Para obter uma discussão detalhada e as configurações de codificação no Adobe Media Encoder CS5, consulte Recomendações para codificação de vídeo H.264 para Flash Player 10.1 em dispositivos móveis (http://www.adobe.com/devnet/devices/articles/mobile_video_encoding.html).

Nota: *No iOS, somente o vídeo codificado com os codecs Sorenson Spark e On2 VP6 podem ser reproduzidos usando a classe Video. Você também pode reproduzir vídeo codificado H.264 no reprodutor de vídeo do dispositivo ativando o URL para o vídeo, usando a função* `flash.net.navigateToURL()`*. Você também pode reproduzir vídeo H.264 usando a marcação <video> em uma página html exibida num objeto StageWebView.*

Compatibilidade do Flash Player e do AIR com arquivos de vídeo codificados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Player 7 oferece suporte a arquivos FLV codificados com o codec de vídeo Sorenson™ Spark™. O Flash Player 8 dá suporte a arquivos FLV codificados com o codificador Sorenson Spark ou On2 VP6 no Flash Professional 8. O codec de vídeo On2 VP6 oferece suporte a um canal alfa.

O Flash Player 9.0.115.0 e versões posteriores dão suporte a arquivos derivados do formato de recipiente MPEG-4 padrão. Esses arquivos incluem F4V, MP4, M4A, MOV, MP4V, 3GP e 3G2, se contêm vídeo H.264 ou áudio codificado em HE-AAC v2 ou ambos. O H.264 oferece vídeo de qualidade superior com taxas de bits inferiores se comparadas com o mesmo perfil de codificação no Sorenson ou On2. O HE-AAC v2 é uma extensão do AAC, um formato de áudio padrão definido no padrão de vídeo MPEG-4. O HE-AAC v2 usa as técnicas SBR (Spectral Band Replication) e PS (Parametric Stereo) para aumentar a eficiência da codificação em taxas de bits baixas.

A tabela a seguir lista os codecs suportados. Ela também mostra o formato de arquivo SWF correspondente e as versões do Flash Player e do AIR necessárias para reproduzi-los:

Codec	**Versão do formato de arquivo SWF (versão de publicação mais antiga suportada)**	**Flash Player e AIR (versão mais antiga necessária para reprodução)**
Sorenson Spark	6	Flash Player 6, Flash Lite 3
On2 VP6	6	Flash Player 8, Flash Lite 3. Apenas o Flash Player 8 e versões posteriores oferecem suporte à publicação e à reprodução de vídeo On2 VP6.
H.264 (MPEG-4 Parte 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 Parte 3)	9	Flash Player 9 Update 3, AIR 1.0
Speex (áudio)	10	Flash Player 10, AIR 1.5
Nellymoser	6	Flash Player 6

Noções básicas sobre os formatos de arquivo de vídeo Adobe F4V e FLV

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A Adobe oferece os formatos de arquivo de vídeo F4V e FLV para transmitir conteúdo para o Flash Player e o AIR. Para ver uma descrição completa desses formatos de arquivo de vídeo, consulte www.adobe.com/go/video_file_format_br.

O formato de arquivo de vídeo F4V

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A partir da atualização 3 do Flash Player (9.0.115.0) e do AIR 1.0, o Flash Player e o AIR dão suporte ao formato de vídeo Adobe F4V, que é baseado no formato ISO MP4. Subconjuntos do formato dão suporte a diferentes recursos. O Flash Player espera que um arquivo F4V válido comece com uma das seguintes caixas de nível superior:

- ftyp

  A caixa ftyp identifica os recursos para os quais um programa deve oferecer suporte para executar um determinado formato de arquivo.

- moov

  A caixa moov é o cabeçalho de um arquivo F4V. Ela contém uma ou mais outras caixas que, por sua vez, contêm outras caixas que definem a estrutura dos dados F4V. Um arquivo F4V deve conter somente uma caixa moov.

- mdat

  Uma caixa mdat contém a carga de dados do arquivo F4V. Um arquivo FV contém apenas uma caixa mdat. Também deve haver uma caixa moov no arquivo porque a caixa mdat não pode ser compreendida por si só.

Os arquivos F4V dão suporte a inteiros multibyte na ordem de bytes big-endian, pela qual o byte mais importante ocorre primeiro, no endereço mais baixo.

O formato de arquivo de vídeo FLV

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O formato de arquivo Adobe FLV contém dados de áudio e vídeo codificados para entrega pelo Flash Player. É possível usar um codificador, como o Adobe Media Encoder ou o Sorenson™ Squeeze, para converter um arquivo de vídeo do QuickTime ou do Windows Media em um arquivo FLV.

Nota: *Você pode criar arquivos FLV importando vídeo para o Flash e o exportando como um arquivo FLV. Você pode usar o plug-in de exportação de FLV para exportar arquivos FLV de aplicativos de edição de vídeos suportados. Para carregar arquivos FLV de um servidor Web, registre a extensão do nome de arquivo e o tipo MIME no servidor Web. Consulte a documentação do seu servidor Web. O tipo MIME de arquivos FLV é* `video/x-flv`*. Para obter mais informações, consulte* *"Sobre a configuração de arquivos FLV para hospedagem em um servidor" na página 521.*

Para obter mais informações sobre arquivos FLV, consulte "Tópicos avançados sobre arquivos de vídeo" na página 520.

Vídeo externo versus incorporado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O uso de arquivos de vídeo externos oferece determinados recursos que não estão disponíveis quando você usa vídeo importado:

- Clipes de vídeo mais longos podem ser usados no seu aplicativo sem diminuir a velocidade da reprodução. Arquivos de vídeo externos usam a memória cache, o que significa que arquivos grandes são armazenados em pequenas partes e acessados dinamicamente. Por esse motivo, os arquivos F4V e FLV externos exigem menos memória do que os arquivos de vídeo incorporados.

- Um arquivo de vídeo externo pode ter uma taxa de quadros diferente do que o arquivo SWF no qual ele é reproduzido. Por exemplo, você pode definir a taxa de quadros do arquivo SWF como 30 quadros por segundo (fps) e a taxa de quadros de vídeo como 21 fps. Esta configuração lhe dá mais controle sobre o vídeo do que o vídeo incorporado e assegura a reprodução contínua. Ela também permite reproduzir arquivos de vídeo com taxas de quadro diferentes sem precisar alterar o conteúdo do arquivo SWF existente.
- Com arquivos de vídeo externos, a reprodução do conteúdo SWF não é interrompida enquanto o arquivo de vídeo está sendo carregado. Às vezes, os arquivos de vídeo importados podem interromper a reprodução de documentos para executar certas funções, como acessar uma unidade de CD-ROM. Os arquivos de vídeo podem executar funções independentemente do conteúdo SWF sem interromper a reprodução.
- É mais fácil legendar o conteúdo em vídeo com arquivos FLV externos porque você pode acessar os metadados do vídeo usando manipuladores de eventos.

Noções básicas sobre a classe Video

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Video permite exibir vídeo de fluxo contínuo ao vivo em um aplicativo sem incorporá-lo ao seu arquivo SWF. É possível capturar e reproduzir vídeo ao vivo usando o método `Camera.getCamera()`. Você também pode usar a classe Video para reproduzir arquivos de vídeo por HTTP ou a partir do sistema de arquivos local. Há várias maneiras diferentes de usar a classe Video em seus projetos:

- Carregue um arquivo de vídeo dinamicamente usando as classes NetConnection e NetStream e exiba o vídeo em um objeto Video.
- Capture a entrada da câmera do usuário. Para mais informações, consulte "Trabalhando com câmeras" na página 537.
- Use o componente FLVPlayback.
- Use o controle VideoDisplay.

Nota: *As ocorrências de um objeto Video no Palco são ocorrências da classe Video.*

Embora a classe Video esteja no pacote flash.media, ela herda da classe flash.display.DisplayObject. Portanto, toda a funcionalidade do objeto de exibição, como transformações de matriz e filtros, também se aplica a ocorrências de Video.

Para obter mais informações, consulte "Manipulação de objetos de exibição" na página 175, "Trabalho com geometria" na página 214 e "Filtro de objetos de exibição" na página 274.

Carregamento de arquivos de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O processo de carregar vídeos usando as classes NetStream e NetConnection inclui várias etapas. Como prática recomendada, as etapas para adicionar o objeto Video à lista de exibição, anexando o objeto NetStream à instância Video e chamando o método `play()` do objeto NetStream devem ser realizadas na ordem especificada:

1. Crie um objeto NetConnection. Se você estiver se conectando a um arquivo de vídeo local ou a um arquivo que não está usando um servidor, como o Flash Media Server 2 da Adobe, passe `null` para o método `connect()` para reproduzir arquivos de vídeo de um endereço HTTP ou de uma unidade local. Se você estiver se conectando a um servidor, defina o parâmetro como o URI do aplicativo que contém o arquivo de vídeo no servidor.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
```

2. Crie um novo objeto Video que exiba o vídeo e adicione-o à lista de exibição do estágio, conforme é mostrado no snippet a seguir:

```
var vid:Video = new Video();
addChild(vid);
```

3. Crie um objeto NetStream, enviando o objeto NetConnection como um argumento para o construtor. O snippet a seguir conecta um objeto NetStream à instância NetConnection e configura os manipuladores de evento para o fluxo:

```
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. Anexe o objeto NetStream ao objeto Video usando o método `attachNetStream()` do objeto Video, conforme é mostrado no snippet a seguir:

```
vid.attachNetStream(ns);
```

5. Chame o método `play()` do objeto NetStream com o URL do arquivo de vídeo como argumento para iniciar a reprodução do vídeo. O snippet a seguir carrega e reproduz um arquivo de vídeo com o nome "video.mp4" no mesmo diretório que o arquivo SWF:

```
ns.play("video.mp4");
```

Mais tópicos da Ajuda

Flex: controle Spark VideoPlayer

spark.components.VideoDisplay

Controle da reprodução de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe NetStream oferece quatro principais métodos para controlar a reprodução de vídeos:

`pause()`: pausa a reprodução de um vídeo em fluxo contínuo. Se o vídeo já estiver pausado, chamar este método não terá efeito.

`resume()`: reinicia a reprodução de um vídeo em fluxo contínuo que foi suspenso. Se o vídeo já estiver em reprodução, chamar este método não terá efeito.

`seek()`: busca o quadro-chave mais próximo do local especificado (um deslocamento, em segundos, a partir do início do fluxo).

`togglePause()`: pausa ou reinicia a reprodução de um fluxo contínuo.

Nota: *Não há um método `stop()`. Para interromper um fluxo, você deve pausar a reprodução e fazer a busca até o início do fluxo de vídeo.*

Nota: *O método `play()` não reinicia a reprodução; ele é usado para carregar arquivos de vídeo.*

O exemplo a seguir demonstra como controlar um vídeo usando vários botões diferentes. Para executar o exemplo, crie um novo documento e adicione quatro ocorrências de botão ao seu espaço de trabalho (`pauseBtn`, `playBtn`, `stopBtn` e `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();
}
```

Quando você clica na ocorrência do botão `pauseBtn` durante a reprodução do vídeo, o arquivo de vídeo é pausado. Se o vídeo já estiver pausado, clicar neste botão não terá efeito. Clicar na ocorrência do botão `playBtn` reinicia a reprodução do vídeo se a reprodução foi pausada anteriormente; caso contrário, o botão não terá efeito se o vídeo já estiver sendo reproduzido.

Detecção do final de um fluxo de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para monitorar o começo ou o final de um fluxo de vídeo, você precisa adicionar um ouvinte de eventos à ocorrência de NetStream para monitorar o evento `netStatus`. O seguinte código demonstra como monitorar os diversos códigos durante a reprodução do vídeo:

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void
{
    trace(event.info.code)
}
```

O código anterior gera a seguinte saída:

```
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
```

Os dois códigos que você deseja monitorar especificamente são “NetStream.Play.Start” e “NetStream.Play.Stop”, que sinalizam o início e o fim da reprodução do vídeo. O seguinte snippet usa uma instrução de opção para filtrar esses dois códigos e rastrear uma mensagem:

```
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;
    }
}
```

Ao monitorar o evento `netStatus` (`NetStatusEvent.NET_STATUS`), você pode criar um player de vídeo que carregue o próximo vídeo de uma lista de exibição depois que a reprodução do vídeo atual for concluída.

Reprodução de vídeo no modo de tela cheia

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Player e o AIR permitem criar um aplicativo em tela cheia para a reprodução de vídeos e dão suporte ao dimensionamento de vídeo em tela cheia.

Para conteúdo do AIR em execução em modo de tela cheia na área de trabalho, o protetor de tela do sistema e as opções de economia de energia ficam desativados durante a reprodução até que a entrada de vídeo pare ou que o usuário saia do modo de tela cheia.

Para obter detalhes sobre como usar o modo de tela cheia, consulte “Trabalho com o modo de tela cheia” na página 169.

Ativação do modo de tela cheia do Flash Player em um navegador

Para que você possa implementar o modo de tela cheia do Flash Player em um navegador, ative-o através do modelo Publish do seu aplicativo. Os modelos que permitem o modo de tela cheia incluem as tags `<object>` e `<embed>` que contêm um parâmetro `allowFullScreen`. O exemplo a seguir mostra o parâmetro `allowFullScreen` em uma tag `<embed>`.

```
<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>
```

No Flash, selecione Arquivo -> Configurações de publicação e, na aba HTML da caixa de diálogo Configurações de publicação, selecione o modelo Somente Flash - Permitir tela cheia.

No Flex, verifique se o modelo HTML inclui as tags `<object>` e `<embed>` que dão suporte a tela cheia.

Inicialização do modo de tela cheia

Para conteúdo Flash Player executado em um navegador, você inicia o modo de tela cheia para vídeo em resposta a um clique do mouse ou a um pressionamento de tecla. Por exemplo, você pode iniciar o modo de tela cheia quando o usuário clica em um botão chamado Tela cheia ou seleciona um comando Tela cheia em um menu de contexto. Para responder ao usuário, adicione um ouvinte de eventos ao objeto no qual ocorre a ação. O seguinte código adiciona um ouvinte de eventos a um botão no qual o usuário clica para entrar no modo de tela cheia:

```
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;

}
```

O código inicia o modo de tela cheia definindo a propriedade `Stage.displayState` como `StageDisplayState.FULL_SCREEN`. Este código dimensiona todo o palco em tela cheia, sendo que o vídeo é dimensionado proporcionalmente ao espaço que ocupa no palco.

A propriedade `fullScreenSourceRect` permite especificar uma determinada área do palco a ser dimensionada para tela cheia. Primeiro, defina o retângulo que você deseja dimensionar para tela cheia. Depois, atribua-o à propriedade `Stage.fullScreenSourceRect`. Esta versão da função `fullScreenButtonHandler()` adiciona duas linhas a mais de código que dimensionam somente o vídeo para tela cheia.

```
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;
}
```

Embora este exemplo chame um manipulador de eventos em resposta a um clique do mouse, a técnica de entrar no modo de tela cheia é a mesma para o Flash Player e o AIR. Defina o retângulo que você deseja dimensionar e configure a propriedade `Stage.displayState`. Para obter mais informações, consulte Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

O exemplo completo, mostrado a seguir, adiciona um código que cria a conexão e o objeto NetStream para o vídeo e começa a reproduzi-lo.

```
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
        }
    }
}
```

A função `onMetaData()` é uma função de retorno de chamada que manipula os metadados do vídeo, se houver. Uma função de retorno de chamada é a função que o tempo de execução chama em resposta a algum tipo de ocorrência ou evento. Neste exemplo, a função `onMetaData()` é um stub que atende ao requisito de fornecer a função. Para obter mais informações, consulte "Criação de métodos de retorno de chamada para metadados e pontos de sinalização" na página 501

Saída do modo de tela cheia

Um usuário pode sair do modo de tela cheia usando um dos atalhos de teclado, como a tecla Escape. Você pode encerrar o modo de tela cheia no ActionScript definindo a propriedade `Stage.displayState` como `StageDisplayState.NORMAL`. O código do exemplo a seguir finaliza o modo de tela cheia quando ocorre o evento NetStream.Play.Stop `netStatus`.

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);

private function netStatusHandler(event:NetStatusEvent)
{
    if(event.info.code == "NetStream.Play.Stop")
        stage.displayState = StageDisplayState.NORMAL;
}
```

Aceleração de hardware em tela cheia

Quando você dimensiona novamente a área retangular do palco para o modo de tela cheia, o Flash Player ou o AIR usa aceleração por hardware caso ela esteja disponível e ativada. O tempo de execução usa o adaptador de vídeo do computador para acelerar o dimensionamento do vídeo ou de uma parte do palco para o tamanho de tela cheia. Nessas circunstâncias, os aplicativos do Flash Player frequentemente se beneficiam alternando da classe Video para a classe StageVideo (ou classe Camera; Flash Player 11.4/AIR 3.4 e superior).

Para obter mais informações sobre aceleração de hardware no modo de tela cheia, consulte "Trabalho com o modo de tela cheia" na página 169. Para obter mais informações sobre StageVideo, consulte "Uso da classe StageVideo para apresentação acelerada por hardware" na página 528.

Arquivos de vídeo em fluxo contínuo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para reproduzir arquivos em fluxo contínuo no Flash Media Server, você pode usar as classes NetConnection e NetStream para se conectar a uma ocorrência do servidor remoto e reproduzir um fluxo especificado. Para especificar um servidor RTMP (Real-Time Messaging Protocol), passe a URL RTMP desejada, como "rtmp://localhost/appName/appInstance", para o método `NetConnection.connect()` em vez de passar `null`. Para reproduzir um determinado fluxo ao vivo ou gravado do Flash Media Server, passe um nome de identificação referente aos dados ao vivo publicados por `NetStream.publish()` ou um nome de arquivo gravado para reprodução ao método `NetStream.play()`.

Envio de vídeo para um servidor

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se você deseja criar aplicativos mais complexos que envolvam objetos de vídeo ou câmera, o Flash Media Server oferece uma combinação de recursos de fluxo de mídia e um ambiente de desenvolvimento para criar e disponibilizar aplicativos de mídia para um grande público. Esta combinação permite que os desenvolvedores criem aplicativos como Vídeo sob Demanda, transmissões ao vivo de eventos pela Web e streaming Mp3, além de publicação de vídeos em blogs, envio de vídeos via mensagens instantâneas e ambientes de bate-papo com recursos multimídia. Para obter mais informações, consulte a documentação on-line do Flash Media Server em www.adobe.com/go/learn_fms_docs_br.

Noções básicas sobre pontos de sinalização

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível incorporar pontos de sinalização a um arquivo de vídeo Adobe F4V ou FLV durante a codificação. Historicamente, os pontos de sinalização eram incorporados a filmes para dar ao projetista um sinal visual que indicava que o rolo de filme estava perto do fim. Nos formatos de vídeo Adobe F4V e FLV, um ponto de sinalização permite que você dê início a uma ou mais ações no aplicativo quando ele ocorrer no fluxo de vídeo.

Você pode usar vários tipos diferentes de pontos de sinalização com vídeos criados no Flash. Você pode usar o ActionScript para interagir com pontos de sinalização que incorpora a um arquivo de vídeo ao criá-lo.

- Pontos de sinalização de navegação: incorpore pontos de sinalização de navegação ao pacote de metadados e fluxo de vídeo quando codificar o arquivo de vídeo. Use pontos de sinalização de navegação para que os usuários possam fazer a busca até uma determinada parte de um arquivo.
- Pontos de sinalização de evento: incorpore pontos de sinalização de evento ao pacote de metadados e fluxo de vídeo quando codificar o arquivo de vídeo. Você pode criar um código para manipular os eventos que são acionados em pontos específicos durante a reprodução do vídeo.

- Pontos de sinalização do ActionScript: os pontos de sinalização do ActionScript só estão disponíveis para o componente FLVPlayback do Flash. Eles são pontos de sinalização externos criados e acessados com código ActionScript. Você pode criar um código para acionar esses pontos de sinalização em relação à reprodução do vídeo. Eles são menos precisos do que os pontos de sinalização incorporados (até um décimo de segundo), porque o player de vídeo os rastreia separadamente. Se você pretende criar um aplicativo em que os usuários poderão navegar até um ponto de sinalização, deve criar e incorporar pontos de sinalização quando codificar o arquivo em vez de usar pontos de sinalização do ActionScript. Você deve incorporar os pontos de sinalização ao arquivo FLV porque eles são mais precisos.

Os pontos de sinalização de navegação criam um quadro-chave no local especificado do ponto de sinalização, por isso você pode usar um código para deslocar o indicador de reprodução de um player de vídeo até esse local. É possível definir pontos específicos em um arquivo de vídeo no qual você deseja que os usuários façam uma busca. Por exemplo, o seu vídeo pode ter vários capítulos ou segmentos, e você pode controlá-lo incorporando pontos de sinalização de navegação ao arquivo de vídeo.

Para obter mais informações sobre como codificar arquivos de vídeo Adobe com pontos de sinalização, consulte "Incorporação de pontos de sinalização" em *Uso do Flash*.

É possível acessar parâmetros de ponto de sinalização criando código ActionScript. Os parâmetros de ponto de sinalização fazem parte do objeto de evento recebido pelo manipulador de retorno de chamada.

Para iniciar certas ações no seu código quando um arquivo FLV atingir um ponto de sinalização específico, use o manipulador de eventos `NetStream.onCuePoint`.

Para sincronizar uma ação para um ponto de sinalização em um arquivo de vídeo F4V, recupere os dados do ponto de sinalização usando as funções de retorno de chamada `onMetaData()` ou `onXMPData()` e acione o ponto de sinalização usando a classe Timer do ActionScript 3.0. Para obter mais informações sobre pontos de sinalização F4V, consulte "Uso de onXMPData()" na página 513.

Para obter mais informações sobre como manipular pontos de sinalização e metadados, consulte "Criação de métodos de retorno de chamada para metadados e pontos de sinalização" na página 501.

Criação de métodos de retorno de chamada para metadados e pontos de sinalização

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível iniciar ações no seu aplicativo quando metadados específicos são recebidos pelo player ou quando se atingem determinados pontos de sinalização. Quando esses eventos ocorrem, você deve usar métodos de retorno de chamada específicos como manipuladores de eventos. A classe NetStream especifica os seguintes eventos de metadados, que podem ocorrer durante a reprodução: `onCuePoint` (somente em arquivos FLV), `onImageData`, `onMetaData`, `onPlayStatus`, `onTextData` e `onXMPData`.

Você deve criar métodos de retorno de chamada para esses manipuladores, ou o tempo de execução do Flash poderá gerar erros. Por exemplo, o código a seguir reproduz um arquivo FLV chamado video.flv na mesma pasta em que reside o arquivo SWF:

```
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);
```

O código anterior carrega um arquivo de vídeo local chamado video.flv e monitora o evento `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`) a ser despachado. Esse evento é despachado quando o código assíncrono nativo gera uma exceção. Nesse caso, ele é despachado quando o arquivo de vídeo contém informações de metadados ou de ponto de sinalização e os ouvintes apropriados não foram definidos. O código anterior manipula o evento `asyncError` e ignora o erro se você não está interessado nas informações de metadados ou de ponto de sinalização do arquivo de vídeo. Se você tivesse um arquivo FLV com metadados e vários pontos de sinalização, a função `trace()` exibiria as seguintes mensagens de erro:

```
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.
```

Os erros ocorrem porque o objeto NetStream não conseguiu encontrar um método de retorno de chamada `onMetaData` ou `onCuePoint`. Existem diversas maneiras de definir esses métodos de retorno de chamada nos seus aplicativos.

Mais tópicos da Ajuda

Flash Media Server: tratamento de metadados em fluxos

Definir a propriedade client do objeto NetStream como Object

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao definir a propriedade `client` como Object ou uma subclasse de NetStream, você pode redirecionar os métodos de retorno de chamada `onMetaData` e `onCuePoint` ou ignorá-los completamente. O exemplo a seguir demonstra como usar um Object vazio para ignorar os métodos de retorno de chamada sem monitorar o evento `asyncError`:

```
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);
```

Para monitorar os métodos de retorno de chamada `onMetaData` ou `onCuePoint`, seria preciso definir métodos para manipular esses métodos de retorno de chamada, conforme mostrado no seguinte snippet:

```
var customClient:Object = new Object();
customClient.onMetaData = metaDataHandler;
function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
```

O código anterior monitora o método de retorno de chamada `onMetaData` e chama o método `metaDataHandler()`, que rastreia uma string. Se o tempo de execução do Flash encontrasse um ponto de sinalização, nenhum erro seria gerado, mesmo que não tenha sido definido um método de retorno de chamada `onCuePoint`.

Criar uma classe personalizada e definir métodos para manipular os métodos de retorno de chamada

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O seguinte código define a propriedade `client` do objeto NetStream como uma classe personalizada, CustomClient, que define manipuladores para os métodos de retorno de chamada:

```
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);
```

Esta é a classe CustomClient:

```
package
{
    public class CustomClient
    {
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
    }
}
```

A classe CustomClient define um manipulador para o manipulador de retorno de chamada `onMetaData`. Se fosse encontrado um ponto de sinalização e o manipulador de retorno de chamada `onCuePoint` fosse chamado, seria despachado um evento `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`) informando que flash.net.NetStream não pôde chamar o manipulador de retorno de chamada onCuePoint. Para evitar esse erro, seria necessário definir um método de retorno de chamada `onCuePoint` na classe CustomClient ou definir um manipulador de eventos para o evento `asyncError`.

Estender a classe NetStream e adicionar métodos para manipular os métodos de retorno de chamada

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O código abaixo cria uma ocorrência da classe CustomNetStream, que é definida em uma listagem de código mais adiante:

```
var ns:CustomNetStream = new CustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

A seguinte listagem de código define a classe CustomNetStream que estende a classe NetStream, manipula a criação do objeto NetConnection necessário e manipula os métodos de manipulador de retorno de chamada `onMetaData` e `onCuePoint`:

```
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");
        }
    }
}
```

Para renomear os métodos `onMetaData()` e `onCuePoint()` na classe CustomNetStream, use o seguinte código:

```
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");
        }
    }
}
```

Estender a classe NetStream e torná-la dinâmica

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível estender a classe NetStream e tornar a subclasse dinâmica para que os manipuladores de retorno de chamada `onCuePoint` e `onMetaData` possam ser adicionados dinamicamente. Isso é demonstrado na listagem a seguir:

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Esta é a classe DynamicCustomNetStream:

```
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);
        }
    }
}
```

Mesmo sem manipuladores para os manipuladores de retorno de chamada `onMetaData` e `onCuePoint`, não são gerados erros porque a classe DynamicCustomNetStream é dinâmica. Para definir métodos para os manipuladores de retorno de chamada `onMetaData` e `onCuePoint`, use o seguinte código:

```
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");
}
```

Definir a propriedade client do objeto NetStream como this

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando se define a propriedade `client` como `this`, o aplicativo procura os métodos `onMetaData()` e `onCuePoint()` no escopo atual. Este exemplo demonstra isso:

```
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);
```

Se os manipuladores de retorno de chamada `onMetaData` ou `onCuePoint` são chamados e não existem métodos para manipular o retorno de chamada, nenhum erro é gerado. Para manipular esses manipuladores de retorno de chamada, crie um método `onMetaData()` e `onCuePoint()` no seu código, como visto no snippet abaixo:

```
function onMetaData(infoObject:Object):void
{
    trace("metadata");
}
function onCuePoint(infoObject:Object):void
{
    trace("cue point");
}
```

Uso de pontos de sinalização e metadados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use os métodos de retorno de chamada NetStream para capturar e processar eventos de metadados e de ponto de sinalização durante a reprodução do vídeo.

Uso de pontos de sinalização

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A tabela a seguir descreve os métodos de retorno de chamada que você pode usar para capturar pontos de sinalização F4V e FLV no Flash Player e no AIR.

Tempo de execução	F4V	FLV
Flash Player 9/AIR1.0		OnCuePoint
		OnMetaData
Flash Player 10		OnCuePoint
	OnMetaData	OnMetaData
	OnXMPData	OnXMPData

O exemplo a seguir usa um loop `for..in` simples para se repetir em cada uma das propriedades do parâmetro `infoObject` que a função `onCuePoint()` recebe. Ele chama a função trace() para exibir uma mensagem quando recebe dados de ponto de sinalização:

```
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]);
    }
}
```

Aparece a seguinte saída:

```
parameters:
name: point1
time: 0.418
type: navigation
```

Este código usa uma das várias técnicas para definir o objeto no qual é executado o método de retorno de chamada. Você pode usar outras técnicas; para obter mais informações, consulte "Criação de métodos de retorno de chamada para metadados e pontos de sinalização" na página 501.

Uso de metadados de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível usar as funções `OnMetaData()` e `OnXMPData()` para acessar as informações de metadados do arquivo de vídeo, inclusive pontos de sinalização.

Uso de OnMetaData()

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os metadados incluem informações sobre o arquivo de vídeo, como duração, largura, altura e taxa de quadros. As informações dos metadados que são adicionadas ao arquivo de vídeo dependem do software utilizado para codificar o arquivo.

```
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]);
    }
}
```

O código anterior gera uma saída parecida com esta:

```
width: 320
audiodelay: 0.038
canSeekToEnd: true
height: 213
cuePoints: ,,
audiodatarate: 96
duration: 16.334
videodatarate: 400
framerate: 15
videocodecid: 4
audiocodecid: 2
```

Se o seu vídeo não tem áudio, as informações de metadados relacionadas a áudio (como `audiodatarate`) retornam `undefined`, pois nenhuma informação de áudio é adicionada aos metadados durante a codificação.

No código anterior, as informações de ponto de sinalização não eram exibidas. Para exibir os metadados de ponto de sinalização, use esta função, que exibe os itens recursivamente em um Object:

```
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);
        }
    }
}
```

O uso do snippet de código anterior para rastrear o parâmetro `infoObject` no método `onMetaData()` gera a seguinte saída:

```
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
```

O exemplo a seguir exibe os metadados de um vídeo MP4. Ele pressupõe que existe um objeto TextArea chamado `metaDataOut`, no qual grava os metadados.

```
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;
        }
    }
}
```

A função `onMetaData()` gerou a seguinte saída para esse vídeo:

```
moovposition=731965
height=352
avclevel=21
videocodecid=avc1
duration=2.36
width=704
videoframerate=25
avcprofile=88
trackinfo=[object Object]
```

Uso do objeto de informação

A tabela a seguir mostra os possíveis valores para os metadados de vídeo que são passados à função de retorno de chamada `onMetaData()` no Object por eles recebido:

Parâmetro	Descrição
aacaot	Tipo de objeto de áudio AAC; são suportados os valores 0, 1 ou 2.
avclevel	Número de nível AVC IDC, como 10, 11, 20, 21 e assim por diante.
avcprofile	Número de perfil AVC, como 55, 77, 100 e assim por diante.
audiocodecid	String que indica o codec de áudio (técnica de codificação/decodificação) que foi usado; por exemplo ".Mp3" ou "mp4a"
audiodatarate	Número que indica a taxa de codificação de áudio, em KB por segundo.
audiodelay	Número que indica em que tempo do arquivo FLV está "time 0" do arquivo FLV original. O conteúdo do vídeo precisa ter um pequeno atraso para que o áudio seja sincronizado corretamente.
canSeekToEnd	Um valor booliano que é `true` se o arquivo FLV está codificado com um quadro-chave no último quadro, o que permite fazer a busca até o final de um arquivo de vídeo com download progressivo. É `falso` se o arquivo FLV não está codificado com um quadro-chave no último quadro.
cuePoints	Uma matriz de objetos, uma para cada ponto de sinalização incorporado no arquivo FLV. O valor é indefinido se o arquivo FLV não contiver pontos de sinalização. Cada objeto tem as seguintes propriedades: • `type`: string que especifica o tipo do ponto de sinalização como "navegação" ou "evento". • `name`: string que é o nome do ponto de sinalização. • `time`: número que é a hora do ponto de sinalização em segundos, com uma precisão de três casas decimais (milésimos de segundo). • `parameters`: objeto opcional que tem pares de nome/valor que o usuário designa ao criar os pontos de sinalização.
duration	Um número que representa a duração do arquivo de vídeo em segundos.
framerate	Um número que representa a taxa de quadros do arquivo FLV.
height	Um número que representa a altura do arquivo FLV em pixels.
seekpoints	Uma matriz que lista os quadros-chave disponíveis como carimbos de data/hora em milésimos de segundo. Opcional.
tags	Uma matriz de pares chave/valor que representam as informações no núcleo "ilst", que é o equivalente de tags ID3 para arquivos MP4. O iTunes usa essas tags. Pode ser usado para exibir arte-final, se disponível.
trackinfo	Objeto que fornece informações sobre todas as faixas do arquivo MP4, inclusive o ID de descrição da amostra.
videocodecid	Uma string que representa a versão de codec que foi usada para codificar o vídeo. - por exemplo, "avc1" ou "VP6F"

Parâmetro	Descrição
videodatarate	Um número que representa a taxa de dados do vídeo do arquivo FLV.
videoframerate	Taxa de quadros do vídeo MP4.
width	Um número que representa a largura do arquivo FLV em pixels.

A tabela a seguir mostra os valores possíveis para o parâmetro `videocodecid`:

videocodecid	Nome do codec
2	Sorenson H.263
3	Vídeo de tela (somente SWF versão 7 e posteriores)
4	VP6 (somente SWF versão 8 e posteriores)
5	Vídeo VP6 com canal alfa (somente SWF versão 8 e posteriores)

A tabela a seguir mostra os valores possíveis para o parâmetro `audiocodecid`:

audiocodecid	Nome do codec
0	não compactado
1	ADPCM
2	Mp3
4	Nellymoser @ 16 kHz mono
5	Nellymoser, 8 kHz mono
6	Nellymoser
10	AAC
11	Speex

Uso de onXMPData()

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

A função de retorno de chamada `onXMPData()` recebe informações específicas da plataforma XMP (Extensible Metadata Platform) da Adobe que são incorporadas ao arquivo de vídeo Adobe F4V ou FLV. Os metadados XMP incluem pontos de sinalização e outros metadados de vídeo. O suporte para metadados XMP foi introduzido com o Flash Player 10 e o Adobe AIR 1.5 e é suportado pelas versões subsequentes.

Este exemplo processa dados de ponto de sinalização nos metadados XMP:

```
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);
        }
    }
}
```

No caso de um arquivo de vídeo curto chamado startrekintro.f4v, este exemplo produz as linhas de rastreamento a seguir. As linhas mostram os dados de ponto de sinalização referentes a pontos de sinalização de navegação e de eventos nos metadados XMP:

```
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
```

Nota: *Nos dados XMP, o tempo é armazenado como Tiques DVA e não como segundos. Para calcular o tempo do ponto de sinalização, divida a hora inicial pela taxa de quadros. Por exemplo, a hora inicial de 7695905817600 dividida por uma taxa de quadros de 254016000000 é igual a 30:30.*

Para ver os metadados XMP brutos completos, que incluem a taxa de quadros, remova os identificadores de comentário (//'s) que precedem a segunda e a terceira instruções `trace()` no início da função `onXMPData()`.

Para obter mais informações sobre a XMP, consulte:

- partners.adobe.com/public/developer/xmp/topic.html
- www.adobe.com/devnet/xmp/

Uso de metadados de imagem

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O evento `onImageData` envia dados de imagem como uma matriz de bytes por um canal de dados AMF0. Os dados podem estar nos formatos JPEG, PNG ou GIF. Defina um método de retorno de chamada `onImageData()` para processar essas informações, da mesma forma que você definiria métodos de retorno de chamada para `onCuePoint` e `onMetaData`. Este exemplo acessa e exibe dados de imagem usando um método de retorno de chamada `onImageData()`:

```
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);
}
```

Uso de metadados de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O evento `onTextData` envia dados de texto por meio de um canal de dados AMF0. Os dados de texto estão no formato UTF-8 e contêm informações adicionais sobre formatação com base na especificação de texto com tempo 3GP. Essa especificação define um formato de legenda padronizado. Defina um método de retorno de chamada `onTextData()` para processar essas informações, da mesma forma que você definiria métodos de retorno de chamada para `onCuePoint` ou `onMetaData`. No próximo exemplo, o método `onTextData()` exibe o número de identificação da faixa e o texto da faixa correspondente.

```
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);
}
```

Como monitorar a atividade do NetStream

Flash Player 10.3 e posterior, Adobe AIR 2.7 e posterior

É possível monitorar a atividade do NetStream para coletar informações necessárias ao suporte de análise e relatório de utilização de mídia. Os recursos de monitoramento abordados nesta seção permitem criar bibliotecas de avaliação de mídia que coletam dados sem uma paridade próxima com o reprodutor de vídeo específico que reproduz a mídia. Isso permite que os desenvolvedores do seu cliente escolham seus reprodutores de vídeo favoritos ao utilizar a biblioteca. Utilize a classe NetMonitor para monitorar a criação e a atividade de objetos NetStream num aplicativo. A classe NetMonitor fornece uma lista de NetStreams ativos existentes e também despacha um evento sempre que um objeto NetStream é criado.

Um objeto NetStream despacha os eventos listados na seguinte tabela, dependendo do tipo de mídia que está sendo reproduzida:

Evento	Download progressivo	Fluxo contínuo RTMP	Fluxo contínuo HTTP
NetStream.Play.Start	Sim	Sim	Não
NetStream.Play.Stop	Sim	Sim	Não
NetStream.Play.Complete	Sim	Sim	Não
NetStream.SeekStart.Notify	Sim	Sim	Sim
NetStream.Seek.Notify	Sim	Sim	Sim
NetStream.Unpause.Notify	Sim	Sim	Sim
NetStream.Unpause.Notify	Sim	Sim	Sim
NetStream.Play.Transition	Não aplicável	Sim	Não aplicável
NetStream.Play.TransitionComplete	Não aplicável	Sim	Não aplicável
NetStream.Buffer.Full	Sim	Sim	Sim
NetStream.Buffer.Flush	Sim	Sim	Sim
NetStream.Buffer.Empty	Sim	Sim	Sim

O objeto NetStreamInfo associado a uma instância do NetStream também armazena os últimos metadados e objetos de dados XMP que foram encontrados na mídia.

Quando a mídia é exibida por meio de um fluxo contínuo HTTP, os eventos NetStream.Play.Start, NetStream.Play.Stop e NetStream.Play.Complete não são despachados, uma vez que o aplicativo tem pleno controle do fluxo de mídia. Um reprodutor de vídeo deve sintetizar e despachar esses eventos para os fluxos HTTP.

Do mesmo modo, os eventos NetStream.Play.Transition e NetStream.Play.TransitionComplete não são despachados para download progressivo ou para mídia HTTP. A comutação dinâmica de taxa de bits é um recurso do RTMP. Se um reprodutor de vídeo que utilize fluxo HTTP suportar um recurso semelhante, o reprodutor poderá sintetizar e despachar eventos de transição.

Mais tópicos da Ajuda

Central do Desenvolvedor da Adobe: Como medir o consumo de vídeo no Flash

Como monitorar eventos do NetStream

Os tipos de eventos fornecem dados de utilização valiosos: `netStatus` e `mediaTypeData`. Além disso, um temporizador pode ser utilizado para registrar periodicamente a posição da cabeça de reprodução do NetStream.

Os eventos `netStatus` fornecem informações que você pode utilizar para determinar a quantidade do fluxo contínuo que foi vista pelo usuário. Os eventos de transição de buffer e fluxo contínuo RTMFP também produzem um evento `netStatus`.

Os eventos `mediaTypeData` fornecem informações de metadados e dados XMP. O evento Netstream.Play.Complete é despachado como um evento `mediaTypeData`. Outros dados integrados no fluxo também estão disponíveis mediante eventos `mediaTypeData`, incluindo pontos de sinalização, texto e imagens.

O exemplo a seguir mostra como criar uma classe que monitora eventos de status e de dados a partir de quaisquer NetStreams ativos num aplicativo. Geralmente, esse tipo de classe atualiza os dados que tinha interesse em analisar para um servidor com o fim de coleta.

```
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
        }
    }

  }
}
```

Como detectar o domínio do reprodutor

O URL e o domínio da página da web na qual um usuário exibe o conteúdo de mídia nem sempre estão disponíveis de imediato. Se for permitido pelo site da web que faz a hospedagem, você poderá utilizar a classe ExternalInterface para obter o URL exato. No entanto, alguns sites da web que permitem reprodutores de vídeo de terceiros não permitem a utilização de interfaces externas (ExternalInterface). Nesses casos, você pode obter o domínio da página da web atual a partir da propriedade `pageDomain` da classe Security. O URL completo não é divulgado para preservar a segurança e a privacidade do usuário.

O domínio da página está disponível na propriedade estática `pageDomain` da classe Security:

```
var domain:String = Security.pageDomain;
```

Tópicos avançados sobre arquivos de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os tópicos a seguir abordam algumas questões especiais sobre o trabalho com arquivos FLV.

Sobre a configuração de arquivos FLV para hospedagem em um servidor

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando você trabalha com arquivos FLV, pode ser necessário configurar o servidor para trabalhar com o formato de arquivo FLV. MIME (Multipurpose Internet Mail Extensions) é uma especificação de dados padronizada que permite enviar arquivos não ASCII por conexões de Internet. Navegadores da Web e clientes de email são configurados para interpretar inúmeros tipos MIME de maneira que possam enviar e receber vídeo, áudio, gráficos e texto formatado. Para carregar arquivos FLV de um servidor Web, talvez você precise registrar a extensão de arquivo e o tipo MIME no servidor Web; para isso, consulte a documentação do servidor Web. O tipo MIME de arquivos FLV é `video/x-flv`. As informações completas sobre o tipo de arquivo FLV são as seguintes:

- Tipo MIME: video/x-flv
- Extensão de arquivo: .flv
- Parâmetros necessários: nenhum
- Parâmetros opcionais: nenhum
- Considerações de codificação: os arquivos FLV são binários; alguns aplicativos podem exigir que o subtipo application/octet-stream seja definido
- Problemas de segurança: nenhum
- Especificação publicada: www.adobe.com/go/video_file_format_br

A Microsoft mudou a forma de tratamento de fluxos de mídia no servidor Web Microsoft Internet Information Services (IIS) 6.0 em relação a versões anteriores. As versões anteriores do IIS não exigem modificações para transmitir vídeo Flash em fluxo. No IIS 6.0, o servidor Web padrão fornecido com o Windows 2003, o servidor requer um tipo MIME para reconhecer que os arquivos FLV são fluxo de mídia.

Quando arquivos SWF que transmitem arquivos FLV externos em fluxo contínuo são colocados no Microsoft Windows Server® 2003 e visualizados em um navegador, o arquivo SWF é reproduzido corretamente, mas o vídeo FLV não é transmitido em fluxo contínuo. Este problema afeta todos os arquivos FLV colocados no Windows Server 2003, inclusive os arquivos criados com versões anteriores da Ferramenta de autoria do Flash e com o Macromedia Flash Video Kit para Dreamweaver MX 2004 da Adobe. Esses arquivos funcionarão corretamente se você testá-los em outros sistemas operacionais.

Para obter informações sobre como configurar o Microsoft Windows 2003 e o Microsoft IIS Server 6.0 para transmissão de vídeos FLV em fluxo contínuo, consulte www.adobe.com/go/tn_19439_br.

Sobre direcionamento de arquivos FLV locais no Macintosh

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se você tentar reproduzir um arquivo FLV local de uma unidade que não é do sistema em um computador com o Apple® Macintosh® usando um caminho que contém uma barra relativa (/), o vídeo não será reproduzido. Unidades que não são do sistema incluem mas não se limitam a CD-ROMs, discos rígidos particionados, mídia de armazenamento removíveis e dispositivos de armazenamento conectados.

Nota: *O motivo desta falha é uma limitação do sistema operacional e não do Flash Player ou do AIR.*

Para que um arquivo FLV seja reproduzido de uma unidade que não é do sistema em um Macintosh, faça referência a ela com um caminho absoluto usando uma notação baseada em dois pontos (:) em vez de barra (/). A lista a seguir mostra a diferença entre os dois tipos de notação:

- Notação baseada em barra**:** myDrive/myFolder/myFLV.flv

- Notação baseada em dois pontos**:** (Mac OS®) myDrive:myFolder:myFLV.flv

Exemplo de vídeo: jukebox de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo a seguir cria uma jukebox de vídeo simples que carrega uma lista de vídeos dinamicamente para reprodução em sequência. Isso permite que você crie um aplicativo com o qual um usuário pode navegar por uma série de tutoriais em vídeo ou talvez especificar quais anúncios devem ser reproduzidos antes de reproduzir o vídeo solicitado pelo usuário. Este exemplo mostra os seguintes recursos do ActionScript 3.0:

- Atualização de um indicador de reprodução com base no progresso da reprodução de um arquivo de vídeo
- Monitoramento e análise dos metadados de um arquivo de vídeo
- Manipulação de códigos específicos em um fluxo de rede
- Carregamento, reprodução, pausa e interrupção de um FLV carregado dinamicamente
- Redimensionamento de um objeto de vídeo na lista de exibição com base nos metadados do fluxo de rede

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Jukebox de vídeo podem ser encontrados na pasta Samples/VideoJukebox. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
VideoJukebox.fla ou VideoJukebox.mxml	O arquivo principal do aplicativo para Flex (MXML) ou Flash (FLA).
VideoJukebox.as	A classe que fornece a funcionalidade principal do aplicativo.
playlist.xml	Um arquivo que lista quais arquivos de vídeo serão carregados na jukebox de vídeo.

Carregamento de um arquivo externo de lista de reprodução de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O arquivo externo playlist.xml especifica os vídeos que devem ser carregados e a ordem em que serão reproduzidos. Para carregar o arquivo XML, você precisa usar um objeto URLLoader e um objeto URLRequest, como visto neste código:

```
uldr = new URLLoader();
uldr.addEventListener(Event.COMPLETE, xmlCompleteHandler);
uldr.load(new URLRequest(PLAYLIST_XML_URL));
```

Este código é colocado no construtor da classe VideoJukebox para que o arquivo seja carregado antes da execução de qualquer outro código. Assim que termina o carregamento do arquivo XML, é chamado o método `xmlCompleteHandler()`, que analisa o arquivo externo em um objeto XML, como visto neste código:

```
private function xmlCompleteHandler(event:Event):void
{
    playlist = XML(event.target.data);
    videosXML = playlist.video;
    main();
}
```

O objeto XML `playlist` contém o XML bruto do arquivo externo, enquanto o XML do vídeo é um objeto XMLList que contém apenas os nós de vídeo. Um arquivo playlist.xml de exemplo pode ser visto no seguinte snippet:

```
<videos>
    <video url="video/caption_video.flv" />
    <video url="video/cuepoints.flv" />
    <video url="video/water.flv" />
</videos>
```

Para finalizar, o método `xmlCompleteHandler()` chama o método `main()`, que configura as várias ocorrências de componente na lista de exibição, bem como os objetos NetConnection e NetStream que são usados para carregar os arquivos FLV externos.

Criação da interface de usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para criar a interface de usuário, você precisa arrastar cinco ocorrências de Button até a lista de exibição e dar a elas os seguintes nomes de ocorrência: `playButton`, `pauseButton`, `stopButton`, `backButton` e `forwardButton`.

Para cada uma dessas ocorrências de Button, você terá de atribuir um manipulador para o evento `click`, como visto no seguinte snippet:

```
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);
```

O método `buttonClickHandler()` usa uma instrução de opção para determinar qual ocorrência de Button foi clicada, como visto neste código:

```
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;
    }
}
```

Em seguida, adicione uma ocorrência de Slider à lista de exibição e dê a ela um nome de ocorrência de `volumeSlider`. O seguinte código define a propriedade `liveDragging` da ocorrência de Slider como `true` e um ouvinte de eventos para o evento `change` da ocorrência de Slider:

```
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);
```

Adicione uma ocorrência de ProgressBar à lista de exibição e dê a ela um nome de ocorrência de `positionBar`. Defina a propriedade `mode` como manual, conforme visto no seguinte snippet:

```
positionBar.mode = ProgressBarMode.MANUAL;
```

Para terminar, adicione uma ocorrência de Label à lista de exibição e dê a ela um nome de ocorrência de `positionLabel`. O valor dessa ocorrência de Label será definido pela ocorrência de Timer.

Monitoramento dos metadados de um objeto de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o Flash Player encontra metadados de cada um dos vídeos carregados, o manipulador de retorno de chamada `onMetaData()` é chamado na propriedade `client` do objeto NetStream. O seguinte código inicializa um Object e configura o manipulador de retorno de chamada especificado:

```
client = new Object();
client.onMetaData = metadataHandler;
```

O método `metadataHandler()` copia seus dados para a propriedade meta definida anteriormente no código. Isso permite que você acesse os metadados do vídeo atual a qualquer momento em todo o aplicativo. Em seguida, o objeto de vídeo no Palco é redimensionado para corresponder às dimensões retornadas nos metadados. Por último, a ocorrência da barra de progresso positionBar é movida e redimensionada com base no tamanho do vídeo que está sendo reproduzido. O seguinte código contém o método `metadataHandler()` inteiro:

```
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;
}
```

Carregamento dinâmico de um vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para carregar cada um dos vídeos dinamicamente, o aplicativo usa um objeto NetConnection e NetStream. O código a seguir cria um objeto NetConnection e passa `null` para o método `connect()`. Especificando `null`, o Flash Player se conecta a um vídeo do servidor local em vez de se conectar a um servidor, como o Flash Media Server.

O seguinte código cria ocorrências de NetConnection e de NetStream, define um ouvinte de eventos para o evento `netStatus` e atribui Object `client` à propriedade `client`:

```
nc = new NetConnection();
nc.connect(null);

ns = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
ns.client = client;
```

O método `netStatusHandler()` é chamado sempre que o status do vídeo é alterado. Isso inclui as ocasiões em que a reprodução de um vídeo é iniciada ou interrompida, quando o vídeo é armazenado em buffer ou quando um fluxo de vídeo não é encontrado. O seguinte código lista o evento `netStatusHandler()`:

```
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.
    }
}
```

O código anterior avalia a propriedade de código do objeto de informações e filtra se o código é “NetStream.Play.Start”, “NetStream.Play.StreamNotFound” ou “NetStream.Play.Stop”. Todos os demais códigos serão ignorados. Se o fluxo de rede estiver iniciando, o código iniciará a ocorrência de Timer que atualiza o indicador de reprodução. Se o fluxo de rede não for encontrado ou se for interrompido, a ocorrência de Timer será interrompida e o aplicativo tentará reproduzir o próximo vídeo da lista de reprodução.

Sempre que Timer é executado, a ocorrência da barra de progresso `positionBar` atualiza sua posição atual chamando o método `setProgress()` da classe ProgressBar, e a ocorrência de Label `positionLabel` é atualizada com o tempo decorrido e o total de tempo do vídeo atual.

```
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.
    }
}
```

Controle do volume do vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível controlar o volume do vídeo carregado dinamicamente definindo a propriedade `soundTransform` no objeto NetStream. O aplicativo de jukebox de vídeo permite modificar o nível de volume alterando o valor da ocorrência de Slider `volumeSlider`. Este código mostra como alterar o nível de volume atribuindo o valor do componente Slider a um objeto SoundTransform, que é definido com a propriedade `soundTransform` no objeto NetStream:

```
private function volumeChangeHandler(event:SliderEvent):void
{
    volumeTransform.volume = event.value;
    ns.soundTransform = volumeTransform;
}
```

Controle da reprodução de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O restante do aplicativo controla a reprodução de vídeos quando o vídeo chega ao final do fluxo ou quando o usuário acessa o vídeo anterior ou o próximo.

O seguinte método recupera a URL do vídeo no objeto XMLList relativo ao índice que está selecionado:

```
private function getVideo():String
{
    return videosXML[idx].@url;
}
```

O método `playVideo()` chama o método `play()` no objeto NetStream para carregar o vídeo que está selecionado:

```
private function playVideo():void
{
    var url:String = getVideo();
    ns.play(url);
}
```

O método `playPreviousVideo()` reduz o índice de vídeos atual, chama o método `playVideo()` para carregar o novo arquivo de vídeo e define a barra de progresso como visível:

```
private function playPreviousVideo():void
{
    if (idx > 0)
    {
        idx--;
        playVideo();
        positionBar.visible = true;
    }
}
```

O último método, `playNextVideo()`, aumenta o índice de vídeos e chama o método `playVideo()`. Se o vídeo atual é o último da lista de reprodução, o método `clear()` é chamado no objeto Video, e a propriedade `visible` da ocorrência da barra de progresso é definida como `false`:

```
private function playNextVideo():void
{
    if (idx < (videosXML.length() - 1))
    {
        idx++;
        playVideo();
        positionBar.visible = true;
    }
    else
    {
        idx++;
        vid.clear();
        positionBar.visible = false;
    }
}
```

Uso da classe StageVideo para apresentação acelerada por hardware

Flash Player 10.2 e posterior

Flash Player otimiza o desempenho de vídeo utilizando a aceleração por hardware para decodificação de H.264. É possível melhorar ainda mais o desempenho utilizando a API StageVideo. O vídeo do palco permite que seu aplicativo aproveite a apresentação acelerada por hardware.

Tempos de execução compatíveis com a API StageVideo incluem:

- Flash Player 10.2 e posterior

Nota: *No Flash Player 11.4/AIR 3.4 e superior, é possível usar entrada da câmera com StageVideo.*

O código-fonte disponível para download e detalhes adicionais do recurso de vídeo do palco estão disponíveis em Introdução ao Stage Video.

Para um tutorial rápido de início do StageVideo, consulte Trabalhando com o Stage Video.

Sobre aceleração por hardware usando StageVideo

A apresentação acelerada por hardware – que inclui dimensionamento de vídeo, conversão de cor e blitting – aumenta os benefícios de desempenho da codificação acelerada por hardware. Em dispositivos que oferecem aceleração da GPU (hardware), você pode usar um objeto flash.media.StageVideo para processar vídeo diretamente no hardware do dispositivo. O processamento direto libera a CPU para executar outras tarefas enquanto a GPU lida com o vídeo. A classe herdada Video, por outro lado, normalmente usa a apresentação por software. A apresentação por software ocorre na CPU e pode consumir uma parcela significativa dos recursos do sistema.

Atualmente poucos dispositivos oferecem aceleração completa de GPU. Contudo, o vídeo do palco permite aos aplicativos aproveitar ao máximo qualquer aceleração por hardware disponível.

A classe StageVideo não torna a classe Video obsoleta. Funcionando juntas, essas duas classes proporcionam a experiência de exibição de vídeo otimizada permitida pelos recursos do dispositivo a qualquer momento específico. Seu aplicativo aproveita a aceleração de hardware ouvindo os eventos apropriados e alternando entre StageVideo e Video, conforme necessário.

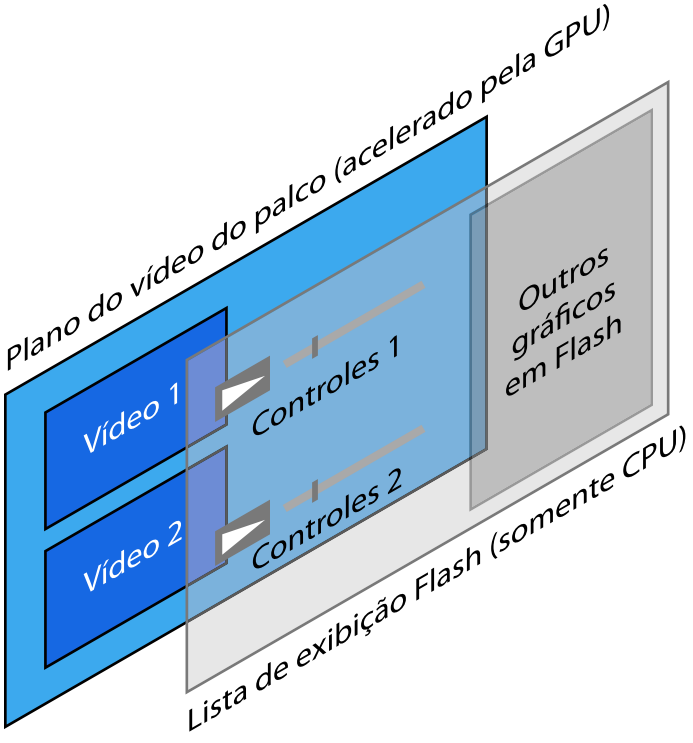
A classe StageVideo impõe certas restrições sobre o uso de vídeo. Antes de implementar StageVideo, reveja as orientações e verifique se sua aplicação pode aceitá-lo. Se você aceitar as restrições, use a classe StageVideo sempre que o Flash Player detectar que a apresentação acelerada por hardware está disponível. Consulte "Orientações e limitações" na página 530.

Planos paralelos: lista de exibição do Flash e Stage Video

Com o modelo de vídeo do palco, o Flash Player pode separar o vídeo da lista de exibição. O Flash Player divide o visor composto entre dois planos ordenados Z:

Plano do vídeo do palco O plano do vídeo do palco fica em segundo plano. Exibe somente vídeo acelerado por hardware. Devido a esse design, esse plano não fica disponível se a aceleração por hardware não for compatível ou não estiver disponível no dispositivo. No ActionScript, objetos StageVideo manipulam vídeos reproduzidos no plano de vídeo do palco.

Plano de lista de exibição Flash As entidades da lista de exibição Flash são compostas em um plano na frente do plano do vídeo do palco. As entidades da lista de exibição incluem qualquer coisa que o tempo de execução renderizar, inclusive controles de reprodução. Quando a aceleração de hardware não está disponível, vídeos podem ser reproduzidos somente nesse plano, usando o objeto da classe Video. O vídeo do palco é sempre exibido atrás dos gráficos da lista de exibição Flash.



Planos de exibição de vídeo

O objeto StageVideo é exibido em uma região não rotacionada e retangular da tela alinhada com a janela. Você não pode estender em camadas objetos atrás do plano de vídeo do palco. Contudo, é possível usar o plano de lista de exibição do Flash para criar camadas com outros gráficos sobre o objeto de vídeo do palco. O vídeo do palco é executado simultaneamente com a lista de exibição do Assim, você pode usar os dois mecanismos juntos para criar um efeito visual unificado que usa dois planos discretos. Por exemplo, pode usar o plano frontal para os controles de reprodução que operam no vídeo do palco em execução em segundo plano.

Vídeo do palco e codec H.264

Em aplicativos Flash Player, a implementação da aceleração por hardware de vídeo envolve duas etapas:

1 Codificação do vídeo como H.264

2 Implementação da API StageVideo

Para melhores resultados, realize as duas etapas. O codec H.264 permite que você aproveite ao máximo a aceleração por hardware, desde a decodificação de vídeo até a apresentação.

O vídeo do palco elimina a releitura de GPU para GPU. Em outras palavras, a GPU não envia frames decodificados de volta para a CPU para composição com objetos da lista de exibição. Em vez disso, a GPU imprime diretamente na tela frames decodificados e renderizados, trás dos objetos da lista de exibição. Esta técnica reduz o uso de memória e CPU e também proporciona uma melhor fidelidade de pixel.

Mais tópicos da Ajuda

"Noções básicas sobre formatos de vídeo" na página 488

Orientações e limitações

Quando o vídeo é executado no modo de tela cheia, o vídeo do palco está sempre disponível se o dispositivo for compatível com aceleração por hardware. O Flash Player, no entanto, também é executado dentro de um navegador. No contexto de navegador, a configuração `wmode` afeta a disponibilidade do vídeo do palco. Tente usar sempre `wmode="direct"` se desejar usar o vídeo do palco. O vídeo do palco não é compatível com outras configurações `wmode` quando o modo de tela cheia não é usado. Essa restrição significa que, no tempo de execução, o vídeo do palco pode vacilar de maneira imprevisível entre estar ou não disponível. Por exemplo, se o usuário sair do modo de tela cheia enquanto o vídeo estiver em execução, o contexto de vídeo inverterá para o navegador. Se o parâmetro `wmode` do navegador não for definido como `"direct"`, o vídeo do palco pode ficar indisponível subitamente. O Flash Player comunica alterações de contexto de reprodução para aplicativos através de um conjunto de eventos. Se você implementar a API StageVideo, mantenha um objeto de vídeo como backup quando o vídeo do palco ficar indisponível.

Por causa dessa relação direta com o hardware, o vídeo do palco restringe alguns recursos de vídeo. O vídeo do palco reforça as seguintes restrições:

- Para cada arquivo SWF, o Flash Player limita a quatro o número de objetos StageVideo que podem exibir vídeo simultaneamente. Entretanto, o limite real pode ser mais baixo, dependendo dos recursos de hardware no dispositivo.
- A temporização do vídeo não é sincronizada com a temporização do conteúdo exibido pelo tempo de execução.
- A área de exibição de vídeo pode ser somente um retângulo. Não é possível usar áreas de exibição mais avançadas, como formas elípticas ou irregulares.
- Não é possível girar o vídeo.
- Não é possível armazenar o vídeo no cache de bitmap nem usar um objeto BitmapData para acessá-lo.
- Não é possível aplicar filtros ao vídeo.
- Não é possível aplicar transformações de cores ao vídeo.
- Não é possível aplicar um valor alfa ao vídeo.
- Os modos de combinação que você aplica a objetos no plano da lista de exibição não se aplicam ao vídeo do palco.
- É possível posicionar o vídeo somente em limites de pixel cheios.

- Embora a renderização da GPU seja a melhor disponível para o hardware de dispositivo específico, ela não é 100% idêntica em termos de pixel em todos os dispositivos. Ligeiras variações ocorrem devido a diferenças de driver e de plataforma.
- Alguns dispositivos não oferecem suporte a todos os espaços de cor necessários. Por exemplo, alguns dispositivos não oferecem suporte a BT.709, o padrão H.264. Nesses casos, você pode usar o BT.601 para acelerar a exibição.
- Não é possível usar o vídeo do palco com configurações WMODE, como normal, opaque ou transparent. O vídeo do palco oferece suporte apenas a `WMODE=direct` quando não está no modo de tela cheia. WMODE não funciona no Safari 4 ou posterior e no IE 9 ou superior.

Na maioria dos casos, essas limitações não afetam aplicativos de player de vídeo. Se puder aceitar estas limitações, use o vídeo do palco sempre que possível.

Mais tópicos da Ajuda

"Trabalho com o modo de tela cheia" na página 169

Uso das APIs StageVideo

O vídeo do palco é um mecanismo no tempo de execução que melhora a reprodução do vídeo e o desempenho do dispositivo. O tempo de execução cria e mantém esse mecanismo. Como desenvolvedor, sua função é configurar seu aplicativo de modo a aproveitá-lo.

Para usar o vídeo do palco, implemente uma estrutura de manipuladores de eventos que detecte quando o vídeo do palco está ou não disponível. Quando você receber uma notificação de que o vídeo do palco está disponível, recupere um objeto StageVideo da propriedade `Stage.stageVideos`. O tempo de execução preenche esse objeto Vector com um ou mais objetos StageVideo. Em seguida, você pode usar um dos objetos StageVideo fornecidos, em vez de um objeto Video, para exibir o fluxo de vídeo.

No Flash Player, quando receber uma notificação de que o vídeo do palco não está mais disponível, alterne seu fluxo de vídeo de volta para um objeto Video.

Nota: *Não é possível criar objetos StageVideo.*

Propriedade Stage.stageVideos

A propriedade `Stage.stageVideos` é um objeto Vector que proporciona acesso a ocorrências de StageVideo. Esse vetor pode conter até quatro objetos StageVideo, dependendo do hardware e dos recursos do sistema. Dispositivos móveis podem estar limitados a um ou nenhum.

Quando o vídeo do palco não está diponível, este vetor não contém nenhum objeto. Para evitar erros no tempo de execução, certifique-se de acessar membros desse vetor somente quando o evento `StageVideoAvailability` mais recente indicar a disponibilidade do vídeo do palco.

Eventos StageVideo

A estrutura da API StageVideo fornece os seguintes eventos:

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY Enviado quando a propriedade `Stage.stageVideos` é alterada. A propriedade `StageVideoAvailabilityEvent.availability` indica se `AVAILABLE` (disponível) ou `UNAVAILABLE` (não disponível). Use este evento para determinar se a propriedade `stageVideos` contém algum objeto StageVideo, em vez de verificar diretamente o comprimento do vetor `Stage.stageVideos`.

StageVideoEvent.RENDER_STATE Enviado quando um objeto NetStream ou Camera tiver sido anexado a um objeto StageVideo e estiver em reprodução. Indica o tipo de decodificação em uso atualmente: hardware, software ou

indisponível (nada é exibido). O destino do evento contém as propriedades `videoWidth` e `videoHeight`que são seguras para serem usadas para redimensionar a janela de exibição de vídeo.

Importante: *As coordenadas obtidas a partir do objeto de destino StageVideo usam as coordenadas Stage visto que não fazem parte da lista de exibição padrão.*

VideoEvent.RENDER_STATE Enviado quando um objeto Video está sendo usado. Indica se a decodificação acelerada por software ou software está em uso. Se este evento indicar a decodificação acelerada por hardware, alterne para um objeto StageVideo, se possível. O destino do evento Video contém as propriedades `videoWidth` e `videoHeight`que são seguras para serem usadas para redimensionar a janela de exibição de vídeo.

Fluxo de trabalho para implementar o recurso StageVideo

Siga estas etapas de nível superior para implementar o recurso StageVideo:

1. `Escute o evento StageVideoAvailabilityEvent. STAGE_VIDEO_AVAILABILITY para descobrir quando o vetor Stage.stageVideos foi alterado.` Consulte "Utilizando o evento StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY" na página 533.
2. Se o evento `StageVideoAvailabilityEvent. STAGE_VIDEO_AVAILABILITY` informar que o vídeo de palco está disponível, use o objeto Vector `Stage.stageVideos` do manipulador de eventos para acessar um objeto StageVideo.
3. Anexe um objeto NetStream utilizando `StageVideo.attachNetStream()` ou anexe um objeto Camera utilizando `StageVideo.attachCamera()`.
4. Reproduza o vídeo usando `NetStream.play()`.
5. Escute o evento `StageVideoEvent.RENDER_STATE` no objeto StageVideo para determinar o status da reprodução do vídeo. O recebimento deste evento também indica que as propriedades de largura e de altura do vídeo foram inicializadas ou alteradas. Consulte "Utilizando os eventos StageVideoEvent.RENDER_STATE e VideoEvent.RENDER_STATE" na página 535.
6. Escute o evento `VideoEvent. RENDER_STATE` no objeto Video. `Este evento fornece os mesmos status que StageVideoEvent. RENDER_STATE; portanto, você também pode usá-lo para determinar se a aceleração de GPU está disponível.` O recebimento deste evento também indica que as propriedades de largura e de altura do vídeo foram inicializadas ou alteradas. Consulte "Utilizando os eventos StageVideoEvent.RENDER_STATE e VideoEvent.RENDER_STATE" na página 535.

Inicializando os ouvintes de evento do StageVideo

Configure seus ouvintes StageVideoAvailabilityEvent e VideoEvent durante a inicialização do aplicativo. Por exemplo, você pode inicializar esses ouvintes no manipulador de eventos `flash.events.Event.ADDED_TO_STAGE`. Este evento garante que seu aplicativo fique visível no palco:

```
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);
        //...
    }
```

Utilizando o evento StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY

No manipulador `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`, decida se deve usar um objeto Video ou StageVideo, com base na disponibilidade de StageVideo. Se a propriedade `StageVideoAvailabilityEvent.availability` estiver definida como `StageVideoAvailability.AVAILABLE`, use StageVideo. Nesse caso, você pode contar com o vetor Stage.stageVideos para conter um ou mais objetos StageVideo. Obtenha um objeto StageVideo da propriedade `Stage.stageVideos` e anexe o objeto NetStream para esse. Porque os objetos StageVideo sempre aparecem no fundo, remova o objeto Video existente (sempre em primeiro plano). Aproveite este manipulador do evento para adicionar um ouvinte para o evento `StageVideoEvent.RENDER_STATE`.

Se a propriedade `StageVideoAvailabilityEvent.availability` estiver definida como `StageVideoAvailability.UNAVAILABLE`, não use o StageVideo nem acesse o vetor `Stage.stageVideos`. Neste caso, anexe o objeto NetStream para um objeto de Video. Finalmente, adicione o objeto StageVideo ou Video ao palco e chame `NetStream.play()`.

O código a seguir mostra como manipular o evento `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`:

```
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);
    }
}
```

Importante: *A primeira vez que um aplicativo acessa o elemento do vetor em Stage.stageVideos[0], o retângulo padrão é configurado para 0,0,0,0, e as propriedades pan e zoom usam os valores padrão. Sempre definir esses valores para suas configurações de preferência. É possível usar as propriedades* `videoWidth` *e* `videoHeight` *do destino dos eventos* `StageVideoEvent.RENDER_STATE` *ou* `VideoEvent.RENDER_STATE` *para calcular as dimensões da janela de exibição de vídeo.*

Baixe todo o código-fonte deste exemplo de aplicativo de Introdução ao vídeo do palco.

Utilizando os eventos StageVideoEvent.RENDER_STATE e VideoEvent.RENDER_STATE

Os objetos StageVideo e Video enviam eventos que informam aos aplicativos quando o ambiente de exibição é alterado. Estes eventos são `StageVideoEvent.RENDER_STATE` e `VideoEvent.RENDER_STATE`.

Um objeto StageVideo ou Vídeo envia um evento de estado renderizado quando um objeto NetStream é anexado e começa a ser reproduzido. Também envia esse evento quando o ambiente de exibição é alterado; por exemplo, quando a janela de exibição de vídeo é redimensionada. Use essas notificações para redefinir sua janela de exibição para os valores atuais de `videoHeight` e `videoWidth` do objeto de destino do evento.

Os estados renderizados informados incluem:

- `RENDER_STATUS_UNAVAILABLE`
- `RENDER_STATUS_SOFTWARE`
- `RENDER_STATUS_ACCELERATED`

Estados renderizados indicam quando a decodificação acelerada por hardware está em uso, independentemente da classe que esteja reproduzindo o vídeo no momento. Marque a propriedade `StageVideoEvent.status` para saber se a decodificação necessária está disponível. Se essa propriedade estiver definida como "indisponível", o objeto StageVideo não pode reproduzir o vídeo. Este estado exige que você coloque imediatamente de novo o objeto NetStream para um objeto Video. Outros estados informam para seu aplicativo as condições de renderização atuais.

A tabela a seguir descreve as implicações de todos os valores do estado de renderização para objetos StageVideoEvent e VideoEvent em Flash Player:

	VideoStatus.ACCELERATED	**VideoStatus.SOFTWARE**	**VideoStatus.UNAVAILABLE**
StageVideoEvent	Tanto a decodificação quanto a apresentação acontecem no hardware. (Melhor desempenho.)	Apresentação no hardware, decodificação no software. (Desempenho aceitável.)	Não há recursos da GPU disponíveis para lidar com o vídeo e nada é exibido. **Retroceder para um objeto Video.**
VideoEvent	Apresentação no software, decodificação no hardware. (Desempenho aceitável apenas em um sistema moderno de desktop. Desempenho degradado em tela cheia.)	Apresentação no software, decodificação no software. (Pior em termos de desempenho de caixa. Desempenho degradado em tela cheia.)	(N/D)

Espaços de cor

O vídeo do palco usa recursos de hardware subjacentes para oferecer suporte a espaços de cor. O conteúdo SWF pode fornecer metadados que indicam seu espaço de cor preferencial. No entanto, o hardware gráfico do dispositivo determina se o espaço de cor pode ser usado. Um dispositivo pode oferecer suporte a vários espaços de cor, enquanto outros não oferecem a nenhum. Se o hardware não oferecer suporte ao espaço de cor necessário, o Flash Player tenta localizar a correspondência mais próxima entre os espaços de cor com suporte.

Para consultar a quais espaços de cor o hardware oferece suporte, use a propriedade `StageVideo.colorSpaces`. Essa propriedade retorna a lista de espaços de cor com suporte em um vetor String:

```
var colorSpace:Vector.<String> = stageVideo.colorSpaces();
```

Para saber qual espaço de cor o vídeo em reprodução no momento está usando, verifique a propriedade `StageVideoEvent.colorSpace`. Verifique essa propriedade em seu manipulador de eventos quanto ao evento `StageVideoEvent.RENDER_STATE`:

```
var currColorSpace:String;

//StageVideoEvent.RENDER_STATE event handler
private function stageVideoRenderState(event:Object):void
{
    //...
    currColorSpace = (event as StageVideoEvent).colorSpace;
    //...
}
```

Se o Flash Player não encontrar um substituto para o espaço de cor incompatível, o vídeo do palco usará o espaço de cor padrão BT.601. Por exemplo, fluxos de vídeo com codificação H.264 normalmente usam o espaço de cor BT.709. Se o hardware do dispositivo não oferecer suporte a BT.709, a propriedade `colorSpace` retornará `"BT601"`. O valor `"unknown"` de `StageVideoEvent.colorSpace` indica que o hardware não oferece nenhum modo de consultar o espaço de cor.

Se seu aplicativo considerar o espaço de cor atual inaceitável, você pode optar por alternar de um objeto StageVideo para um objeto Video. A classe Video oferece suporte a todos os espaços de cor por meio da composição do software.

Capítulo 26: Trabalhando com câmeras

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma câmera conectada ao computador de um usuário pode funcionar como fonte de dados de vídeo, que você pode exibir e manipular usando o ActionScript. A classe Camera é o mecanismo criado no ActionScript para trabalhar com um computador ou dispositivo de câmera.

Nos dispositivos móveis, você também pode usar a classe CameraUI. A classe CameraUI ativa um aplicativo de câmera para permitir que o usuário capture uma tomada de imagem ou de vídeo. Quando o usuário terminar, seu aplicativo poderá acessar a imagem ou o vídeo por meio de um objeto MediaPromise.

Mais tópicos da Ajuda

Christian Cantrell: How to use CameraUI in a Cross-platform Way (Como Usar a Interface do Usuário da Câmara em Várias Plataformas)

Michaël CHAIZE: Android, AIR and the Camera (Android, AIR e a Câmara)

Noções básicas sobre a classe Camera

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O objeto Camera permite que você se conecte à câmera local do usuário e transmita o vídeo localmente (para o usuário) ou remotamente para um servidor (como o Flash Media Server).

Usando a classe Camera, é possível acessar os seguintes tipos de informações sobre a câmera do usuário:

- Quais câmeras instaladas no computador ou dispositivo do usuário estão disponíveis.
- Se há uma câmera instalada
- Se o Flash Player tem acesso à câmera do usuário ou não
- Qual câmera está ativa
- A largura e a altura do vídeo que está sendo capturado

A classe Camera inclui vários métodos e propriedades úteis para trabalhar com objetos Camera. Por exemplo, a propriedade estática `Camera.names` contém uma matriz de nomes de câmeras que estão instaladas no computador do usuário. Você também pode usar a propriedade `name` para exibir o nome da câmera que está ativa.

Nota: *Ao transmitir fluxo de vídeo da câmera pela rede, trate sempre das interrupções da rede. Interrupções de rede podem ocorrer por muitos motivos, especialmente em dispositivos móveis.*

Exibição do conteúdo da câmera na tela

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A conexão a uma câmera pode exigir menos código do que usar as classes NetConnection e NetStream para carregar um vídeo. A classe Camera também pode rapidamente se tornar complicada porque, com o Flash Player, você precisa da permissão do usuário para se conectar à câmera dele antes de acessá-la.

Este código demonstra como usar a classe Camera para se conectar à câmera local de um usuário:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Nota: *A classe Camera não tem um método construtor. Para criar uma nova ocorrência de Camera, use o método estático* `Camera.getCamera()`.

Desenvolvimento do aplicativo de câmera

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao criar um aplicativo que se conecta à câmera de um usuário, considere os seguintes aspectos no seu código:

- Verifique se o usuário tem uma câmera instalada. Trate o caso em que não há nenhuma câmera disponível.
- Somente no Flash Player, verifique se o usuário permitiu acesso à câmera explicitamente. Por motivo de segurança, o player exibe a caixa de diálogo Configurações do Flash Player, onde o usuário pode conceder ou negar acesso à sua câmera. Isso impede que o Flash Player se conecte à câmera de um usuário e transmita um fluxo de vídeo sem a devida permissão. Se um usuário clicar em permitir, o aplicativo poderá se conectar à câmera. Se ele clicar em negar, o aplicativo não conseguirá ter acesso à câmera do usuário. Seus aplicativos sempre devem manipular os dois casos normalmente.
- Somente para o AIR, verifique se a classe Camera conta com o suporte dos perfis de dispositivo compatíveis com seu aplicativo.
- A classe Camera não conta com suporte em navegadores móveis.
- A classe Camera não conta com suporte em aplicativos AIR móveis que usam o modo de renderização pela GPU.
- Em dispositivos móveis, somente uma câmera pode ficar ativa de cada vez.

Mais tópicos da Ajuda

Christophe Coenraets: jogo da velha em vídeo com vários usuários

Mark Doherty: aplicativo Android Radar (fonte)

Conexão à câmera de um usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A primeira etapa para se conectar à câmera de um usuário é criar uma nova ocorrência de Camera criando uma variável do tipo Camera e a inicializando com o valor de retorno do método estático `Camera.getCamera()`.

A próxima etapa é criar um novo objeto de vídeo e conectar o objeto Camera a ele.

A terceira etapa é adicionar o objeto de vídeo à lista de exibição. Você precisa executar as etapas 2 e 3 porque a classe Camera não estende a classe DisplayObject, por isso não é possível adicioná-la diretamente à lista de exibição. Para exibir o vídeo capturado da câmera, crie um novo objeto de vídeo e chame o método `attachCamera()`.

O seguinte código mostra estas três etapas:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Observe que, se o usuário não tiver uma câmera instalada, o aplicativo não exibirá nada.

Na prática, você precisa executar outras etapas que envolvem o seu aplicativo. Para obter mais informações, consulte "Verificação da instalação das câmeras" na página 539 e "Detecção de permissões de acesso à câmera" na página 540.

Mais tópicos da Ajuda

Lee Brimelow: como acessar a câmera em dispositivos Android

Verificação da instalação das câmeras

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Antes de tentar usar qualquer um dos métodos ou propriedades em uma ocorrência de Camera, é recomendável verificar se o usuário tem uma câmera instalada. Há duas maneiras de fazer essa verificação:

- Verifique a propriedade estática `Camera.names` que contém uma matriz de nomes de câmeras que estão disponíveis. Normalmente, essa matriz terá no máximo uma string, porque é provável que a maioria dos usuários não tenha mais de uma câmera instalada. Este código demonstra como verificar a propriedade `Camera.names` para saber se o usuário tem alguma câmera disponível:

  ```
  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.");
  }
  ```

- Verifique o valor de retorno do método estático `Camera.getCamera()`. Se não houver nenhuma câmera disponível ou instalada, o método retornará `null`; caso contrário, ele retornará uma referência a um objeto Camera. Este código demonstra como verificar o método `Camera.getCamera()` para saber se o usuário tem alguma câmera disponível:

```
var cam:Camera = Camera.getCamera();
if (cam == null)
{
    trace("User has no cameras installed.");
}
else
{
    trace("User has at least 1 camera installed.");
}
```

Como a classe Camera não estende a classe DisplayObject, não pode ser adicionada diretamente à lista de exibição usando o método `addChild()`. Para exibir o vídeo capturado da câmera, é preciso criar um novo objeto Video e chamar o método `attachCamera()` na ocorrência de Video.

Este snippet mostra como conectar a câmera, se disponível; se não houver uma câmera, o aplicativo não exibirá nada:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
```

Câmeras de dispositivos móveis

A classe Camera não é suportada no tempo de execução do Flash Player em navegadores móveis.

Nos aplicativos AIR em dispositivos móveis você pode acessar a câmera ou câmeras no dispositivo. Em dispositivos móveis, você pode usar tanto a câmera frontal quanto a câmera traseira, mas somente uma saída de câmera pode ser exibida em um determinado momento. (Conectar uma segunda câmera desconecta a primeira.) A câmera frontal está refletida horizontalmente no iOS; no Android, não está.

Detecção de permissões de acesso à câmera

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Na caixa de proteção do aplicativo do AIR, o aplicativo pode acessar qualquer câmera sem a permissão do usuário. No Android, no entanto, um aplicativo deve especificar a permissão CAMERA do Android no descritor do aplicativo.

Para que o Flash Player possa exibir a saída de uma câmera, o usuário deve permitir o acesso do Flash Player à câmera explicitamente. Quando o método `attachCamera()` é chamado, o Flash Player exibe a caixa de diálogo Configurações do Flash Player, que pede para o usuário conceder ou negar acesso à câmera e ao microfone para o Flash Player. Se o usuário clicar no botão Permitir, o Flash Player exibirá a saída da câmera na ocorrência de Video no Palco. Se o usuário clicar no botão Negar, o Flash Player não conseguirá se conectar à câmera, e o objeto Video não exibirá nada.

Para detectar se o usuário concedeu acesso à câmera para o Flash Player, você pode monitorar o evento `status` da câmera (`StatusEvent.STATUS`), como visto neste código:

```
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"
}
```

A função `statusHandler()` é chamada assim que o usuário clica em Permitir ou Negar. Você pode detectar em qual botão o usuário clicou usando um destes dois métodos:

- O parâmetro `event` da função `statusHandler()` contém uma propriedade de código que inclui a string "Camera.Muted" ou "Camera.Unmuted". Se o valor é "Camera.Muted", isso significa que o usuário clicou no botão Negar e o Flash Player não pode acessar a câmera. Este snippet mostra um exemplo disso:

```
function statusHandler(event:StatusEvent):void
{
    switch (event.code)
    {
        case "Camera.Muted":
            trace("User clicked Deny.");
            break;
        case "Camera.Unmuted":
            trace("User clicked Accept.");
            break;
    }
}
```

- A classe Camera contém uma propriedade somente leitura chamada `muted` que especifica se o usuário negou acesso à câmera (`true`) ou se permitiu acesso a ela (`false`) no painel Privacidade do Flash Player. Este snippet mostra um exemplo disso:

```
function statusHandler(event:StatusEvent):void
{
    if (cam.muted)
    {
        trace("User clicked Deny.");
    }
    else
    {
        trace("User clicked Accept.");
    }
}
```

Verificando o eventos de status a ser despachado, você pode criar um código que lide com a aceitação ou a negação do acesso à câmera pelo usuário e fazer as devidas exclusões. Por exemplo, se o usuário clicar no botão Negar, você poderá exibir uma mensagem informando que ele deve clicar em Permitir se deseja participar de um bate-papo com vídeo, ou você poderá verificar se o objeto Video da lista de exibição foi excluído para liberar recursos do sistema.

No AIR, um objeto Camera não despacha eventos de status, visto que a permissão para usar a câmera não é dinâmica.

Maximizando a qualidade de vídeo da câmera

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Por padrão, as novas ocorrências da classe Video têm 320 pixels de largura por 240 pixels de altura. Para maximizar a qualidade dos vídeos, sempre verifique se o objeto de vídeo tem as mesmas dimensões do vídeo retornado pelo objeto de câmera. Para obter a largura e a altura do objeto de câmera, use as propriedades `width` e `height` da classe Camera; em seguida, você pode definir as propriedades `width` e `height` do objeto de vídeo de modo que correspondam às dimensões dos objetos de câmera, ou pode passar a largura e a altura da câmera para o método construtor da classe Video, como visto neste snippet:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video(cam.width, cam.height);
    vid.attachCamera(cam);
    addChild(vid);
}
```

Uma vez que o método `getCamera()` retorna uma referência a um objeto de câmera (ou `null` se não houver câmeras disponíveis), você poderá acessar os métodos e as propriedades da câmera mesmo que o usuário negue acesso a ela. Isso permite que você defina o tamanho da ocorrência de vídeo usando a altura e a largura nativas da câmera.

```
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);
}
```

Para obter informações sobre o modo de tela cheia, consulte a seção Modo de tela cheia em "Configuração de propriedades do palco" na página 167.

Monitorando o status da câmera

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Camera contém várias propriedades que permitem monitorar o status atual do objeto Camera. Por exemplo, este código exibe diversas propriedades da câmera usando um objeto Timer e uma ocorrência de campo de texto na lista de exibição:

```
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");
}
```

A cada 1/10 de segundo (100 milissegundos) o evento `timer` do objeto Timer é despachado, e a função `timerHandler()` atualiza o campo de texto na lista de exibição.

Capítulo 27: Uso de gerenciamento de direitos digitais

Flash Player 10.1 e posterior, Adobe AIR 1.5 e posterior

O Adobe® Access™ é uma solução de proteção de conteúdo. Ele permite que proprietários de conteúdo, distribuidores e anunciantes obtenham novas fontes de renda ao fornecer acesso direto a conteúdo premium. Os editores usam o Adobe Access para codificar conteúdos, criar políticas e emitir licenças. O Adobe Flash Player e o Adobe AIR incorporam uma biblioteca DRM, o módulo Adobe Access. Este módulo permite que o runtime se comunique com o servidor de licença do Adobe Access e reproduza o conteúdo protegido. O runtime completa o ciclo de vida do conteúdo protegido pelo Adobe Access e distribuídos pelo Flash Media Server.

Com o Adobe Access, provedores de conteúdo podem fornecer conteúdo gratuito e premium. Por exemplo, um consumidor deseja ver um programa de televisão sem anúncios. O consumidor se registra e paga ao provedor de conteúdo. O Consumidor pode então inserir suas credenciais de usuário para ganhar acesso e reproduzir o programas sem os anúncios.

Em outro exemplo, um consumidor deseja ver conteúdo offline enquanto viaja sem acesso a Internet. O fluxo de trabalho offline é suportado em aplicativos AIR. Depois de se registrar e pagar ao provedor pelo conteúdo, o usuário pode acessar e baixar o conteúdo e o aplicativo do AIR associado do website do provedor. Utilizando o aplicativo do AIR, o usuário pode ver o conteúdo offline durante o período permitido. O conteúdo também pode ser compartilhado com outros dispositivos no mesmo grupo de dispositivo usando domínios (**Novo na versão 3.0**).

O Adobe Access também suporta acesso anônimo, que não requer autenticação do usuário. Por exemplo, um provedor pode utilizar acesso anônimo para distribuir conteúdo com anúncios. O acesso anônimo também possibilita ao editor permitir o acesso gratuito ao conteúdo atual por um número específico de dias. O provedor de conteúdo também pode especificar e restringir o tipo e a versão do player necessário para seu conteúdo.

Quando um usuário tenta reproduzir conteúdo protegido no Flash Player ou no Adobe AIR, seu aplicativo deve chamar as APIs de DRM. As APIs de DRM iniciam o fluxo de trabalho para reprodução de conteúdo protegido. O runtime faz contato com o servidor de licenças pelo módulo do Adobe Access. O servidor de licença autentica o usuário, se necessário, e emite uma licença para permitir a reprodução de conteúdo protegido. O tempo de execução recebe a licença e descriptografa o conteúdo para reprodução.

A descrição de como habilitar seu aplicativo para reprodução de conteúdo protegido pelo Adobe Access é apresentada aqui. Não é necessário entender como criptografar conteúdo ou manter políticas utilizando o Adobe Access. No entanto, pressupõe-se que você esteja se comunicando com o servidor de licenças do Adobe Access para autenticar o usuário e recuperar a licença. Também pressupõe-se que você esteja projetando um aplicativo para reproduzir especificamente conteúdo protegido pelo Adobe Access.

Para obter uma visão geral do Adobe Access, incluindo a criação de políticas, consulte a documentação incluída com o Adobe Access.

Mais tópicos da Ajuda

Pacote flash.net.drm

flash.net.NetConnection

flash.net.NetStream

Entendendo o fluxo de trabalho de conteúdo protegido.

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

Importante: o Flash Player 11.5 e superior integram o módulo do Adobe Access, portanto, a etapa de atualização (chamar `SystemUpdater.update(SystemUpdaterType.DRM)`) é desnecessária. Isso inclui os seguintes navegadores e plataformas:

- Controle ActiveX do Flash Player 11.5, para todas as plataformas exceto o Internet Explorer no Windows 8 em processadores Intel
- Plug-in do Flash Player 11.5, para todos os navegadores
- Adobe AIR (desktop e dispositivos móveis)

Isso significa que a etapa de atualização é *necessária* nos seguintes casos:

- Internet Explorer no Windows 8 em processadores Intel
- Flash Player 11.4 e inferior, exceto no Chrome 22 e superior (todas as plataformas) ou 21 e superior (Windows)

Nota: *Ainda é possível chamar `SystemUpdater.update(SystemUpdaterType.DRM)` com segurança em um sistema com Flash Player 11.5 ou superior, mas nenhum item será baixado.*

O seguinte fluxo de trabalho mostra como um aplicativo pode salvar e executar o conteúdo protegido. O fluxo de trabalho pressupõe que o aplicativo é projetado para reproduzir especificamente conteúdo protegido pelo Adobe Access:

1. Obtenha os metadados do conteúdo.
2. Trata das atualizações do Flash Player, se necessário.
3. Verifique se uma licença está disponível localmente. Se sim, carregue-o e siga para o passo 7. Se não, vá para o passo 4.
4. Verifique se a autenticação é necessária. Se não, vá para o passo 7.
5. Caso a autenticação seja necessária, obtenha as credenciais de autenticação para o usuário e passe-as para o servidor de licenças.
6. Caso seja necessário o registro de domínio, cadastre-se no domínio (AIR 3.0 e superior).
7. Uma vez que a autenticação aconteça, baixe a licença do servidor.
8. Reproduza o conteúdo.

Se não tiver ocorrido nenhum erro e o usuário foi autorizado com êxito a exibir o conteúdo, o objeto NetStream enviará um objeto DRMStatusEvent. O aplicativo inicia a reprodução. O objeto DRMStatusEvent guarda as informações de voucher relacionadas, que identificam a política e as permissões do usuário. Por exemplo, ele possui informações sobre se o conteúdo pode ser disponibilizado offline ou quando a licença expira. O aplicativo pode usar esses dados para informar ao usuário sobre o status de sua política. Por exemplo, o aplicativo pode exibir o número de dias restantes que o usuário possui para exibir o conteúdo em uma barra de status.

Caso o usuário tenha permissão para o acesso offline, o voucher será armazenado em cache e o conteúdo criptografado será transferido para o computador do usuário. O conteúdo torna-se acessível para a duração definida na duração do armazenamento em cache da licença. A propriedade `detail no evento contém "DRM.voucherObtained"`. O aplicativo decide onde armazenar o conteúdo localmente para que ele esteja disponível offline. Você também pode pré-carregar vouchers usando a classe DRMManager.

Nota: *O armazenamento em cache e o pré-carregamento dos vouchers é suportado no AIR e no Flash Player. No entanto, o download e o armazenamento de conteúdo criptografado são suportados somente no AIR.*

É de responsabilidade do aplicativo lidar especificamente com as ocorrências de erros. Estes eventos incluem casos nos quais o usuário insere credenciais válidas, mas o voucher que protege o conteúdo criptografado restringe o acesso ao conteúdo. Por exemplo, um usuário autenticado não pode acessar o conteúdo se os direitos não estiverem pagos. Este caso também pode ocorrer quando dois integrantes registrados do mesmo provedor tentar compartilhar conteúdo adquirido somente por um deles. O aplicativo deve informar o usuário do erro e oferecer uma sugestão alternativa. Uma sugestão alternativa comum são as instruções sobre como registrar e pagar por direitos de visualização.

Fluxo de trabalho detalhado

Flash Player 10.1 e posterior, AIR 2.0 e posterior

Este fluxo de trabalho fornece uma visão mais detalhada do fluxo de trabalho do conteúdo protegido. Esse fluxo de trabalho descreve as APIs especificas utilizadas para reproduzir conteúdo protegido pelo Adobe Access.

1. Utilizando um objeto URLLoader, carregue os bytes do arquivo de metadados do conteúdo protegido. Defina este objeto para uma variável como `metadata_bytes`.

   Todo o conteúdo controlado pelo Adobe Access possui metadados do Adobe Access. Quando o conteúdo é empacotado, esses metadados podem ser salvos como um arquivo metadados separado (.metadata) junto com o conteúdo. Para obter mais informações, consulte a documentação do Adobe Access.

2. Crie uma ocorrência de DRMContentData. Coloque este código em um bloco try-catch:

   ```
   new DRMContentData(metadata_bytes)
   ```

   onde *`metadata_bytes`* é o objeto URLLoader obtido no passo 1.

3. (Apenas para o Flash Player) O runtime faz uma verificação sobre o módulo do Adobe Access. Caso não seja encontrado, um código de erro IllegalOperationError 3344 ou um código de erro DRMErrorEvent 3343 será lançado.

   Para tratar desse erro, faça o download do módulo Adobe Access utilizando a API SystemUpdater. Depois que o módulo for baixado, os objeto SystemUpdater envia um evento COMPLETE. Inclua um ouvinte de eventos para este evento que volta para o passo 2 quando este evento é enviado. O código a seguir demonstra as etapas:

   ```
   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);
   }
   ```

   Se o próprio player precisa ser atualizado, um evento de status é enviado. Para mais informações sobre como manipular este evento, veja "Procurando um evento de atualização" na página 561.

   Nota: *Nos aplicativos AIR, o instalador AIR manipula a atualização do módulo do Adobe Access e de atualizações do runtime necessárias.*

4. Crie ouvintes para ouvir os eventos DRMStatusEvent e DRMErrorEvent enviados do objeto DRMManager:

   ```
   DRMManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
   DRMManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
   ```

   No ouvinte DRMStatusEvent, verifique se o voucher é válido (não null). No ouvinte DRMErrorEvent, manipule DRMErrorEvents. Veja "Uso da classe DRMStatusEvent" na página 553 e "Uso da classe DRMErrorEvent" na página 558.

5. Carregue o voucher (licença) necessário para reproduzir o conteúdo.

Primeiro, tente carregar uma licença armazenada localmente para reproduzir o conteúdo:

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.LOCAL_ONLY)
```

Depois que o carregamento estiver completo o objeto DRMManager envia `DRMStatusEvent.DRM_Status`.

6 Caso o objeto DRMVoucher não seja null, o voucher é válido. Avançar para a etapa 13.

7 Caso o objeto DRMVoucher seja null, verifique o método de autenticação necessário pela política para este conteúdo. Utilize a propriedade `DRMContentData.authenticationMethod`.

8 Caso o método de autenticação seja `ANONYMOUS`, vá para o passo 13.

9 Caso o método de autenticação seja `USERNAME_AND_PASSWORD`, seu aplicativo precisa fornecer um mecanismo para que o usuário insira as credenciais. Passe estas credenciais para o servidor de licenças para autenticar o usuário:

```
DRMManager.authenticate(metadata.serverURL, metadata.domain, username, password)
```

O DRMManager envia um `DRMAuthenticationErrorEvent` caso a autenticação falhe ou um `DRMAuthenticationCompleteEvent` caso a autenticação funcione. Crie ouvintes para estes eventos.

10 Se o método de autenticação for `UNKNOWN`, um método padrão de autenticação deverá ser usado. Nesse caso, o fornecedor de conteúdo determinou que a autenticação fosse feita no modo fora de banda não usando as APIs do ActionScript 3.0. O procedimento para a autenticação padrão deve produzir um testemunho de autenticação que pode ser transmitido para o método `DRMManager.setAuthenticationToken()`.

11 Caso a autenticação falhe, seu aplicativo deve retornar para o passo 9. Certifique-se de que seu aplicativo tenha um mecanismo para manipular e limitar falhas de autenticação repetidas. Por exemplo, depois de três tentativas, mostre uma mensagem para o usuário indicando que a autenticação falhou e que o conteúdo não pode ser reproduzido.

12 Para utilizar o token armazenado em vez de solicitar que o usuário insira as credenciais, defina o token com o método `DRMManager.setAuthenticationToken()`. Então você pode baixar a licença do servidor de licenças e reproduzir o conteúdo como no passo 8.

13 (opcional) Caso a autenticação proceda, você pode capturar o token de autenticação, que é uma matriz de bytes armazenada na memória. Capture este token com a propriedade `DRMAuthenticationCompleteEvent.token`. Você pode armazenar e utilizar o token de autenticação para que o usuário não precise inserir as credenciais repetidamente para este conteúdo. O servidor de licença determina o período de validade do token de autenticação.

14 Caso a autenticação aconteça, baixe a licença do servidor de licenças.

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.FORCE_REFRESH)
```

Depois que o carregamento estiver completo o objeto DRMManager envia DRMStatusEvent.DRMStatus Ouça este evento e quando ele for enviado, o conteúdo pode ser reproduzido.

15 Reproduzir o vídeo criando um objeto NetStream e chamando o método `play()`:

```
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 e objetos da sessão

Quando os `DRMContentData` são criados, eles são usados como objetos da sessão que se referem ao módulo do Flash Player do DRM. Todas as APIs `DRMManager`que recebem esses `DRMContentData` usarão aquele módulo DRM específico. No entanto, existem 2 APIs `DRMManager` que não utilizam `DRMContentData`. São as seguintes:

1. `authenticate()`
2. `setAuthenticationToken()`

Uma vez que não há `DRMContentData` associados, a execução dessas APIs `DRMManager` utilizará o último módulo DRM do disco. Isso poderá se tornar um problema caso ocorra uma atualização no módulo DRM no meio do fluxo de trabalho do DRM do aplicativo. Considere a seguinte situação:

1. O aplicativo cria um objeto DRMContentData`contentData1`, que usa o *AdobeCP1* como o módulo DRM.
2. O aplicativo executa o método `DRMManager.authenticate(contentData1.serverURL,...)`.
3. O aplicativo executa o método `DRMManager.loadVoucher(contentData1, ...)` .

Caso ocorra uma atualização no módulo DRM antes que o aplicativo consiga chegar à etapa 2, então o método `DRMManager.authenticate()` finalizará a autenticação usando o *AdobeCP2* como o módulo DRM. O método `loadVoucher()` na etapa 3 falhará, uma vez que ele ainda utiliza o*AdobeCP1* como o módulo DRM. A atualização pode ter ocorrido devido a um outro aplicativo que executa a atualização do módulo DRM. É possível evitar essa situação executando a atualização do módulo DRM na inicialização do aplicativo.

Eventos relacionados ao DRM

O tempo de execução envia vários eventos quando um aplicativo tenta reproduzir conteúdo protegido:

- DRMDeviceGroupErrorEvent (somente no AIR), encaminhado pelo DRMManager
- DRMAuthenticateEvent (apenas no AIR), enviado pelo NetStream
- DRMAuthenticationCompleteEvent, enviado pelo DRMManager
- DRMAuthenticationErrorEvent, enviado pelo DRMManager
- DRMErrorEvent, enviado pelo NetStream e pelo DRMManager
- DRMStatusEvent, enviado pelo NetStream e pelo DRMManager
- StatusEvent
- NetStatusEvent. Veja “Procurando um evento de atualização” na página 561

Para oferecer suporte a conteúdo protegido pelo Adobe Access, adicione ouvintes de eventos para manipular eventos de DRM.

Pré-carregamento de vouchers para reprodução offline

Adobe AIR 1.5 e posterior

Você pode pré-carregar os vouchers (licenças) necessários para reproduzir conteúdo protegido pelo Adobe Access. Os vouchers pré-carregados permitem que os usuários visualizem o conteúdo independentemente de terem acesso à Internet. (O processo de pré-carregamento propriamente dito requer conexão com Internet.) Use o método `preloadEmbeddedMetadata()` da classe NetStream e a classe DRMManager do para pré-carregar vouchers. No AIR 2.0 e posteriores, você pode utilizar um objeto DRMContentData para pré-carregar vouchers diretamente. Esta técnica tem preferência porque permite a atualização do objeto DRMContentData independentemente do conteúdo. (O método `preloadEmbeddedData()` extrai DRMContentData do conteúdo.)

Utilizando DRMContentData

Adobe AIR 2.0 e posterior

As seguintes etapas descrevem o fluxo de trabalho para o pré-carregamento do voucher de um arquivo de mídia protegido usando um objeto DRMContentData.

1. Obtenha os metadados binários para o conteúdo empacotado. Caso esteja utilizando o Java Reference Packager do Adobe Access, esse arquivo metadados é automaticamente gerado com uma extensão *.metadata*. Por exemplo, é possível fazer o download destes metadados utilizando a classe URLLoader.
2. Crie um objeto DRMContentData passando os metadados para a função do construtor:

   ```
   var drmData:DRMContentData = new DRMContentData( metadata );
   ```

3. O restante dos passos são idênticos ao fluxo de trabalho do "Entendendo o fluxo de trabalho de conteúdo protegido." na página 545.

Utilizando preloadEmbeddedMetadata()

Adobe AIR 1.5 e posterior

As seguintes etapas descrevem o fluxo de trabalho para o pré-carregamento do voucher de um arquivo de mídia com proteção de DRM usando o `preloadEmbeddedMetadata()`:

1. Baixe e armazene o arquivo de mídia. (Os metadados DRM só podem ser pré-carregados dos arquivos armazenados localmente.)
2. Crie os objetos NetConnection e NetStream, fornecendo implementações para as funções de retorno de chamada `onDRMContentData()` e `onPlayStatus()` do objeto cliente NetStream.
3. Crie um objeto NetStreamPlayOptions e defina a propriedade `stream` para a URL do arquivo de mídia local.
4. Chame o NetStream `preloadEmbeddedMetadata()`, passando no objeto NetStreamPlayOptions que identifica o arquivo de mídia que será analisado.
5. Se o arquivo de mídia contiver metadados DRM, a função de retorno de chamada `onDRMContentData()` será chamada. Os metadados são transmitidos a essa função como objeto DRMContentData.
6. Use o objeto DRMContentData para obter o voucher usando o método `loadVoucher()` do DRMManager.

   Se o valor da propriedade `authenticationMethod` do objeto`DRMContentData` for `flash.net.drm.AuthenticationMethod.USERNAME_AND_PASSWORD`, faça a autenticação do usuário no servidor de direitos de mídia antes de carregar o voucher. As propriedades `serverURL` e `domain` do objeto DRMContentData podem ser passadas para o método `authenticate()` do DRMManager, juntamente com as credenciais do usuário.
7. A função de retorno de chamada `onPlayStatus()` é chamada após a conclusão da análise do arquivo. Se a função `onDRMContentData()` não tiver sido chamada, o arquivo não conterá os metadados necessários para obter um voucher. Essa chamada ausente também pode significar que o Adobe Access não protege este arquivo.

O exemplo de código para o AIR a seguir ilustra como pré-carregar um voucher para um arquivo de mídia local:

```
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();
    }
}
}
```

Membros e eventos relacionados a DRM da classe NetStream

Flash Player 10.1 e posterior, Adobe AIR 1.0 e posterior

A classe NetStream fornece uma conexão de streaming unidirecional entre o Flash Player ou um aplicativo do AIR e o Flash Media Server ou o sistema de arquivos local. (A classe NetStream também suporta o download progressivo.) Um objeto NetStream é um canal em um objeto NetConnection. A classe NetStream envia quatro eventos relacionados ao DRM:

Evento	Descrição
drmAuthenticate (somente AIR)	Definido na classe DRMAuthenticateEvent. Este evento é enviado quando um objeto NetStream tenta reproduzir conteúdo protegido que requer uma credencial de usuário para autenticação antes da reprodução. As propriedades desse evento incluem as propriedades header, usernamePrompt, passwordPrompt e urlPrompt que podem ser usadas para obter e definir as credenciais do usuário. Esse evento ocorre repetidamente até que o objeto NetStream receba credenciais de usuário válidas.
drmError	Definido na classe DRMErrorEvent e enviado quando um objeto NetStream tenta reproduzir conteúdo protegido e encontra um erro relacionado com o DRM. Por exemplo, um objeto de evento de erro DRM é despachado quando a autorização do usuário falha. Este erro pode ocorrer porque o usuário não adquiriu os direitos de visualização do conteúdo. Isto também pode ocorrer porque o provedor de conteúdo não possui suporte para o aplicativo de visualização.
drmStatus	Definido na classe DRMStatusEvent. Este evento é enviado quando o conteúdo protegido começa a ser reproduzido (quando o usuário está autenticado e autorizado a reproduzir o conteúdo. O objeto DRMStatusEvent contém informações relacionadas ao voucher. As informações sobre o voucher incluem se o conteúdo pode ser disponibilizado offline ou quando o voucher expira e o conteúdo não pode mais se exibido.
status	Definido em events.StatusEvent e enviado apenas quando o aplicativo tenta reproduzir conteúdo protegido, chamando o método NetStream.play(). O valor da propriedade do código de status será "DRM.encryptedFLV".

A classe NetStream inclui os seguintes métodos específicos de DRM, para uso no AIR apenas:

Método	Descrição
resetDRMVouchers()	Exclui todos os dados de comprovação de gerenciamento de direitos digitais armazenados localmente em cache. O aplicativo deve baixar o voucher novamente para que o usuário possa acessar o conteúdo criptografado. Por exemplo, o código a seguir remove todos os vouchers do cache: NetStream.resetDRMVouchers();
setDRMAuthenticationCredentials()	Transmite um conjunto de credenciais de autenticação, a saber, nome de usuário, senha e tipo de autenticação, ao objeto NetStream para autenticação. Os tipos de autenticação válidos são `"drm"` e `"proxy"` . Com o tipo de autenticação `"drm"`, as credenciais fornecidas são autenticadas durante o Adobe Access. Com o tipo de autenticação `"proxy"`, as credenciais são autenticadas em comparação ao servidor proxy e devem ser compatíveis às solicitadas por ele. Por exemplo, uma empresa pode requerer que o aplicativo faça a autenticação em um servidor proxy antes que o usuário possa acessar a Internet. A opção do proxy permite este tipo de autenticação. A menos que seja usada a autenticação anônima, depois da autenticação proxy, o usuário ainda precisará ser autenticado comparando ao Adobe Access para obter o voucher e reproduzir o conteúdo. É possível reutilizar `setDRMAuthenticationCredentials()`, com a opção "`drm`" para autenticar durante o Adobe Access.
preloadEmbeddedMetadata()	Analisa se um arquivo de mídia local contém metadados incorporados. Quando metadados relacionados a DRM são encontrados, o AIR chama a função de retorno de chamada `onDRMContentData()` .

Além disso, no AIR, um objeto NetStream chama as funções de retorno de chamada `onDRMContentData()` e `onPlayStatus()` como resultado de uma chamada para o método `preloadEmbeddedMetaData()`. A função `onDRMContentData()` é chamada quando os metadados DRM são encontrados em um arquivo de mídia. A função `onPlayStatus()` é chamada quando o arquivo é analisado. As funções `onDRMContentData()` e `onPlayStatus()` devem ser definidas no objeto `client` atribuído à ocorrência do NetStream. Se você usar o mesmo objeto NetStream para pré-carregar vouchers e reproduzir conteúdo, aguarde pela chamada `onPlayStatus()` gerada pelo `preloadEmbeddedMetaData()` antes de iniciar a reprodução.

No seguinte código do AIR, o nome de usuário ("administrator"), a senha ("password") e o tipo de autenticação "drm" são definidos para a autenticação do usuário. O método setDRMAuthenticationCredentials() precisa fornecer credenciais que correspondam as credenciais conhecidas e aceitas pelo provedor de conteúdo. Estas credenciais são as mesmas credenciais de usuário que permitem a visualização do conteúdo. O código para reproduzir o vídeo e certificar-se de que foi feita uma conexão bem-sucedida ao streaming de vídeo não está incluída aqui.

```
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");
}
```

Uso da classe DRMStatusEvent

Flash Player 10.1, Adobe AIR 1.0 e posterior

Um objeto NetStream envia um objeto DRMStatusEvent quando o conteúdo protegido pelo Adobe Access começa a ser reproduzido com sucesso. (Sucesso implica que a licença foi verificada e que o usuário está autenticado e autorizado a visualizar o conteúdo). DRMStatusEvent também é despachado para usuários anônimos se for permitido acesso a eles. A licença é consultada para verificar se usuários anônimos, que não precisam de autenticação, podem acessar e reproduzir o conteúdo. Usuários anônimos podem ter acesso negado por várias razões. Por exemplo, um usuário anônimo não tem acesso ao conteúdo quando a licença estiver expirada.

O objeto DRMStatusEvent contém informações relacionadas à licença. Tais informações incluem as situações em que a licença pode ser disponibilizada offline ou em que o voucher expira e o conteúdo não pode mais ser visualizado. O aplicativo pode usar esses dados para transmitir suas permissões e o status da política do usuário.

Propriedades DRMStatusEvent

Flash Player 10.1, Adobe AIR 1.0 e posterior

A classe DRMStatusEvent inclui as seguintes propriedades. Algumas propriedades se ficaram disponíveis em versões do AIR posteriores a 1.0. Para informações sobre informações completas, veja *Referência do ActionScript 3.0*.

Para propriedades que não são suportadas no Flash 10.1, a classe DRMVoucher fornece propriedades semelhantes para o Flash Player.

Propriedade	Descrição
contentData	Um objeto DRMContentData contendo os metadados DRM incorporados no conteúdo.
detail (somente AIR)	Uma sequência de caracteres que explica o contexto do evento de status. No DRM 1.0, o único valor válido é DRM.voucherObtained.
isAnonymous (somente AIR)	Indica se o conteúdo, protegido com criptografia pelo Adobe Access, está disponível sem exigir que um usuário forneça credenciais de autenticação (true) ou não (false). Um valor false significa que os usuários devem fornecer um nome de usuário e uma senha que correspondam aos conhecidos e esperados pelo provedor de conteúdo.
isAvailableOffline (apenas no AIR)	Indica se o conteúdo, protegido com criptografia pelo Adobe Access, pode ser disponibilizado offline (true) ou não (false). Para que o conteúdo protegido digitalmente seja disponibilizado offline, seu voucher deve ser armazenado em cache na máquina local do usuário.
isLocal	Indica se o voucher necessário para reproduzir o conteúdo está armazenado em cache localmente.
offlineLeasePeriod (apenas no AIR)	O número restante de dias que o conteúdo pode ser exibido offline.
políticas (somente AIR)	Um objeto personalizado que pode conter propriedades DRM personalizadas.
voucher	O DRMVoucher.
voucherEndDate (somente AIR)	A data absoluta na qual o voucher expira e o conteúdo não pode mais ser exibido.

Criação de um manipulador DRMStatusEvent

Flash Player 10.1, Adobe AIR 1.0 e posterior

O exemplo a seguir cria um manipulador de eventos que fornece as informações de status de conteúdo DRM para o objeto NetStream que originou o evento. Adicione esse manipulador de eventos a um objeto NetStream que aponta para o conteúdo protegido.

```
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
```

Uso da classe DRMAuthenticateEvent

Adobe AIR 1.0 e posterior

O objeto DRMAuthenticateEvent é enviado quando um objeto NetStream tenta reproduzir conteúdo protegido que requer uma credencial de usuário para autenticação antes da reprodução.

O manipulador DRMAuthenticateEvent é responsável pela coleta das credenciais necessárias (nome de usuário, senha e tipo) e pela transmissão dos valores ao método `NetStream.setDRMAuthenticationCredentials()` para validação. Cada aplicativo do AIR deve fornecer algum mecanismo para obter credenciais do usuário. Por exemplo, o aplicativo pode fornecer ao usuário uma interface simples para inserir os valores de nome de usuário e senha. Pode também, fornecer um mecanismo para manipular e limitar tentativas repetidas de autenticação.

Propriedades DRMAuthenticateEvent

Adobe AIR 1.0 e posterior

A classe DRMAuthenticateEvent inclui as seguintes propriedades:

Propriedade	Descrição
authenticationType	Indica se as credenciais fornecidas são para autenticação em comparação ao Adobe Access ("drm") ou um servidor proxy ("proxy"). Por exemplo, a opção "proxy" permite que o aplicativo autentique em um servidor proxy caso seja necessário antes de o usuário acessar a Internet. A menos que seja usada a autenticação anônima, depois da autenticação proxy, o usuário ainda precisará ser autenticado comparando ao Adobe Access para obter o voucher e reproduzir o conteúdo. É possível reutilizar setDRMAuthenticationcredentials(), com a opção "drm" para autenticar durante o Adobe Access.
header	O cabeçalho do arquivo de conteúdo criptografado fornecido pelo servidor. Ele contém informações sobre o contexto do conteúdo criptografado. Esta sequência de caracteres de cabeçalho pode ser enviada ao aplicativo Flash para permitir que o aplicativo crie uma caixa de diálogo com nome de usuário e senha. A sequência de caracteres de cabeçalho pode ser usada como instruções da caixa de diálogo. Por exemplo, o cabeçalho pode ser "Digite o nome de usuário e a senha".
netstream	O objeto NetStream que iniciou esse evento.
passwordPrompt	Um prompt para uma credencial de senha, fornecida pelo servidor. A sequência de caracteres pode incluir instruções para o tipo de senha necessária.
urlPrompt	Um prompt para uma sequência de caracteres de URL, fornecida pelo servidor. A sequência de caracteres pode fornecer o local para onde o nome de usuário e a senha são enviados.
usernamePrompt	Um prompt para uma credencial de nome de usuário, fornecida pelo servidor. A sequência de caracteres pode incluir instruções para o tipo de nome de usuário necessário. Por exemplo, um provedor de conteúdo pode requerer um endereço de email como nome de usuário.

As sequências mencionadas anteriormente são fornecidas somente pelo servidor FMRMS. O servidor do Adobe Access não utiliza essas sequências de caracteres.

Criação de um manipulador DRMAuthenticateEvent

Adobe AIR 1.0 e posterior

O exemplo a seguir cria um manipulador de eventos que transmite um conjunto de credenciais de autenticação codificadas ao objeto NetStream que originou o evento. (O código para reproduzir o vídeo e certificar-se de que foi feita uma conexão bem-sucedida ao streaming de vídeo não está incluída aqui.)

```
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");
}
```

Criação de uma interface para recuperar credenciais do usuário

Adobe AIR 1.0 e posterior

No caso em que o conteúdo protegido requer autenticação do usuário, o aplicativo do AIR normalmente deve obter as credenciais de autenticação do usuário por meio de uma interface de usuário.

A seguir, um exemplo do Flex de uma interface de usuário simples para recuperar credenciais do usuário. Ele consiste de um objeto de painel contendo dois objetos TextInput, um para cada uma das credenciais de nome de usuário e senha. O painel também contém um botão que inicia o método `credentials()`.

```
<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>
```

O método `credentials()` é um método definido pelo usuário que transmite os valores de nome de usuário e senha para o método `setDRMAuthenticationCredentials()`. Depois que os valores são passados, o método `credentials()` redefine os valores dos objetos TextInput.

```
<mx:Script>
    <![CDATA[
        public function credentials():void
        {
            videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
            uName.text = "";
            pWord.text = "";
        }
    ]]>
</mx:Script>
```

Uma forma de implementar este tipo de interface simples é incluir o painel como parte de um estado novo. O novo estado tem origem em um estado base onde o objeto DRMAuthenticateEvent é lançado. O exemplo a seguir contém um objeto VideoDisplay com um atributo de origem que aponta para um arquivo FLV protegido. Neste caso, o método `credentials()` é modificado para também retornar o aplicativo ao estado base. Este método faz isto, enviando as credenciais do usuário e reiniciando os valores de objeto TextInput.

```
<?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>
```

Uso da classe DRMErrorEvent

Flash Player 10.1 e posterior, Adobe AIR 1.0 e posterior

O Adobe Flash e o Adobe AIR enviam um objeto DRMErrorEvent quando um objeto NetStream, que está tentando reproduzir conteúdo protegido, encontra um erro relacionado ao DRM. Caso as credenciais de usuário sejam inválidas em um aplicativo do AIR, o objeto DRMAuthenticateEvent envia sinais repetidamente até que o usuário insira credenciais válidas ou o aplicativo recuse outras tentativas. O aplicativo deve ouvir qualquer evento de erro do DRM para detectar, identificar e manipular os erros relacionados a DRM.

Mesmo com credenciais de usuário válidas, os termos do voucher do conteúdo ainda podem impedir que o usuário visualize o conteúdo criptografado. Por exemplo, o usuário pode ter o acesso negado ao tentar visualizar um conteúdo em um aplicativo não autorizado. Um aplicativo não autorizado é um aplicativo no qual o provedor do conteúdo criptografado não foi validado. Neste caso, um objeto DRMErrorEvent é despachado.

Os eventos de erro também podem ser disparados se o conteúdo estiver corrompido ou se a versão do aplicativo não corresponder às especificações do voucher. O aplicativo deve fornecer um mecanismo apropriado para manipular erros.

Propriedades DRMErrorEvent

Flash Player 10.1 e posterior, Adobe AIR 1.0 e posterior

Para uma lista completa de erros, veja Códigos de erro do tempo de execução na referência do ActionScript 3.0. Os erros relacionados ao DRM começam no erro 3300.

Criação de um manipulador DRMErrorEvent

Flash Player 10.1 e posterior, Adobe AIR 1.0 e posterior

O exemplo a seguir cria um manipulador de eventos para o objeto NetStream que originou o evento. Ele é chamado quando o NetStream encontra um erro ao tentar reproduzir conteúdo protegido. Geralmente, quando um aplicativo encontra um erro, ele executa um número de tarefas de limpeza. Em seguida, ele informa o erro ao usuário e fornece opções para solucionar o problema.

```
private function drmErrorEventHandler(event:DRMErrorEvent):void
{
    trace(event.toString());
}
```

Uso da classe DRMManager

Flash Player 10.1 e posterior, Adobe AIR 1.5 e posterior

Use a classe DRMManager para gerenciar vouchers e sessões do servidor de direitos de mídia em aplicativos.

Gerenciamento de Vouchers (apenas no AIR)

Sempre que um usuário reproduz conteúdo protegido, o tempo de execução obtém e armazena a licença necessária para visualizar o conteúdo. Se o aplicativo salva o arquivo localmente e a licença permite a reprodução offline, o usuário pode visualizar o conteúdo no aplicativo do AIR. A reprodução offline local ocorre com sucesso mesmo se a conexão ao servidor de direitos de mídia não estiver disponível. Utilizando o DRMManager e o método `preloadEmbeddedMetadata()` , você pode pré-armazenar o voucher. O aplicativo não tem que obter a licença necessária para visualizar o conteúdo. Por exemplo, seu aplicativo pode baixar o arquivo de mídia e obter o voucher enquanto o usuário ainda está online.

Para pré-carregar um voucher, use o método `preloadEmbeddedMetadata()` do NetStream para obter um conteúdo do objeto DRMContentData. O objeto DRMContentData contém a URL e o domínio do servidor dos direitos de mídia que pode fornecer a licença e descreve se a autenticação é necessária. Com essas informações, você pode chamar o método `loadVoucher()` do DRMManager para obter o voucher e armazená-lo em cache. O fluxo de trabalho dos vouchers pré-carregados é descrito em mais detalhes em "Pré-carregamento de vouchers para reprodução offline" na página 548.

Gerenciamento de sessão

Você também pode usar o DRMManager para autenticar o usuário para um servidor de direitos de mídia e gerenciar sessões persistentes.

Chame o método `authenticate()` do DRMManager para estabelecer uma sessão com o servidor de direitos de mídia. Quando a autenticação for concluída com êxito, o DRMManager despachará um objeto DRMAuthenticationCompleteEvent. Esse objeto contém um token de sessão. Você pode salvar esse token para estabelecer sessões futuras, de forma que o usuário não precise fornecer suas credenciais de conta. Envie o token para o método `setAuthenticationToken()` para estabelecer uma sessão autenticada. (As configurações do servidor que geraram o sinal determinam a expiração do token e outros atributos. O código do aplicativo do AIR não deve interpretar a estrutura de dados do token, porque a estrutura pode alterar atualizações posteriores do AIR.)

É possível transferir tokens de autenticação para outros computadores. Para proteger os tokens, armazene-os no depósito local criptografado do AIR. Consulte "Armazenamento local criptografado" na página 733 para obter mais informações.

Eventos do DRMStatus

Flash Player 10.1 e posterior, Adobe AIR 1.5 e posterior

O DRMManager despacha um objeto DRMStatusEvent após uma chamada bem-sucedida para o método `loadVoucher()`.

Se um voucher for obtido, a propriedade `detail` (apenas no AIR) do objeto de evento terá um valor: "DRM.voucherObtained", e a propriedade `voucher` conterá o objeto DRMVoucher.

Se um voucher não for obtido, a propriedade `detail` (apenas no AIR) ainda terá o valor: "DRM.voucherObtained"; no entanto, a propriedade `voucher` será `null`. Um voucher não pode ser obtido se, por exemplo, for utilizado LoadVoucherSetting de *localOnly* e não existir um voucher armazenado em cache localmente.

Se a chamada `loadVoucher()` não for concluída com êxito, talvez por causa de um erro de autenticação ou comunicação, o DRMManager despachará um objeto DRMErrorEvent ou DRMAuthenticationErrorEvent em vez disso.

Eventos DRMAuthenticationComplete

Flash Player 10.1 e posterior, Adobe AIR 1.5 e posterior

O DRMManager despacha um objeto DRMAuthenticationCompleteEvent quando um usuário é autenticado com êxito por meio de uma chamada do método `authenticate()`.

O objeto DRMAuthenticationCompleteEvent contém um token reutilizável que pode ser usado para persistir em autenticações do usuário em sessões do aplicativo. Passe esse token para o método `setAuthenticationToken()` do DRMManager, a fim de estabelecer novamente a sessão. (Os atributos do token como, por exemplo, sua expiração, são definidos pelo criador do token. A Adobe não fornece uma API para a examinar atributos de token).

Eventos DRMAuthenticationError

Flash Player 10.1 e posterior, Adobe AIR 1.5 e posterior

O DRMManager despacha um objeto DRMAuthenticationErrorEvent quando um usuário não pode ser autenticado com êxito por meio de uma chamada para os métodos `authenticate()` ou `setAuthenticationToken()`.

Uso da classe DRMContentData

Flash Player 10.1 e posterior, Adobe AIR 1.5 e posterior

O objeto DRMContentData contém as propriedades dos metadados do conteúdo protegido pelo Adobe Access. As propriedades DRMContentData contêm as informações necessárias para obter um comprovante de licença para exibir o conteúdo. Você pode utilizar a classe DRMContentData para obter o arquivo de metadados associado ao conteúdo, como descrito no "Fluxo de trabalho detalhado" na página 546.

Para obter mais informações, consulte a descrição da classe DRMContentData em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Atualização do Flash Player para oferecer suporte ao Adobe Access

Flash Player 10.1 e posterior

Importante: o Flash Player 11.5 e superior integram o módulo do Adobe Access, portanto, a etapa de atualização (chamar `SystemUpdater.update(SystemUpdaterType.DRM)`) é desnecessária. Isso inclui os seguintes navegadores e plataformas:

- Controle ActiveX do Flash Player 11.5, para todas as plataformas exceto o Internet Explorer no Windows 8
- Plug-in do Flash Player 11.5, para todos os navegadores
- Adobe AIR (desktop e dispositivos móveis)

Isso significa que a etapa de atualização é *necessária* nos seguintes casos:

- Internet Explorer no Windows 8
- Flash Player 11.4 e inferior, exceto no Chrome 22 e superior (todas as plataformas) ou 21 e superior (Windows)

Nota: *Ainda é possível chamar* `SystemUpdater.update(SystemUpdaterType.DRM)` *com segurança em um sistema com Flash Player 11.5 ou superior, mas nenhum item será baixado.*

Para oferecer suporte ao Adobe Access o Flash Player requer o módulo do Adobe Access. Quando o Flash Player tenta reproduzir conteúdo protegido, o tempo de execução indica se o módulo ou uma nova versão do Flash Player precisam ser baixados. Desta maneira, o Flash Player fornece aos desenvolvedores SWF a opção de não atualizar caso queiram.

Na maioria dos casos, para reproduzir conteúdo protegido, os desenvolvedores de SWF atualizam para o módulo requerido pelo Adobe Access ou pelo player. Para atualizar, pode-se utilizar a API SystemUpdater para obter a última versão do módulo Adobe Access ou do Flash Player.

A API SystemUpdater permite apenas uma atualização por vez. O código de erro 2202 indica que uma atualização já está em andamento na ocorrência atual do tempo de execução ou em outra ocorrência. Por exemplo, se uma atualização está em andamento em uma ocorrência do Flash Player no Internet Explorer, uma atualização não poderá ser executada em uma ocorrência em execução no Firefox.

A API SystemUpdater é suportada para plataformas de desktop apenas.

Nota: *Em versões do Flash Player anteriores a 10.1, utilize o mecanismo de atualização suportado em versões anteriores do player (baixe e instale manualmente do site www.adobe.com ou ExpressInstall). O instalador do AIR também lida com as atualizações necessárias para o Adobe Access e não suporta a API SystemUpdater.*

Procurando um evento de atualização

Flash Player 10.1 e posterior

Quando uma atualização do módulo Adobe Access é necessária, o objeto NetStream envia um NetStatusEvent com um valor de código `DRM.UpdateNeeded`. Esse valor indica que o objeto NetStream não pode reproduzir o stream protegido com nenhum dos módulos do Adobe Access instalados. Procure por esse evento e chame o seguinte código:

```
SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
```

Esse código atualiza o módulo Adobe Access instalado no player. Não é necessária a aprovação do usuário para essa atualização de módulo.

Caso o módulo Adobe Access não seja encontrado, um erro é exibido. Consulte o passo 3 do "Fluxo de trabalho detalhado" na página 546.

Nota: *Caso play() é chamado em um stream criptografado em players anteriores ao 10.1, um NetStatusEvent como valor de código de NetStream.Play.StreamNotFound é enviado. Em versões anteriores de players, utilize o mecanismo de atualização suportado para estes players (baixe e instale manualmente do site www.adobe.com ou ExpressInstall).*

Quando uma atualização para o próprio player é necessária o objeto SystemUpdater envia um StatusEvent com um código de `DRM.UpdateNeededButIncompatible`. Para uma atualização do player, a aprovação do usuário é necessária. Forneça em seu aplicativo um interface para que o usuário concorde e inicie a atualização do player. Procure pelo evento StatusEvent e chame o seguinte código:

```
SystemUpdater.update(flash.system.SystemUpdaterType.SYSTEM);
```

Este código inicia a atualização do player.

Eventos adicionais da classe SystemUpdater estão documentados em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Depois que a atualização do player estiver completo, o usuário é redirecionado para a página onde a atualização tem início. O módulo Adobe Access é baixado e o stream pode ser reproduzido.

Licenças fora de banda

Flash Player 11 e posterior, Adobe AIR 3.0 e posterior

As licenças também podem ser obtidas fora de banda (sem conectar a um servidor de licença do Adobe Access) mediante o armazenamento do voucher (licença) no disco e na memória, usando o método `storeVoucher`.

Para executar vídeos criptografados no Flash Player e no AIR, o respectivo runtime precisa obter o voucher de DRM para o vídeo. O voucher de DRM contém a chave de decodificação do vídeo e é gerado pelo servidor de licença do Adobe Access que o usuário implantou.

O runtime do Flash Player/AIR geralmente obtém esse voucher enviando uma solicitação do voucher para o servidor de licença do Adobe Access indicado na classe dos metadados DRM do vídeo (classe `DRMContentData`). O aplicativo do Flash/AIR pode acionar a solicitação chamando o método `DRMManager.loadVoucher()`. Ou o tempo de execução do Flash Player/AIR automaticamente solicitará uma licença no início da reprodução do vídeo criptografado, caso não haja nenhuma licença para o conteúdo do disco ou da memória. Em ambos os casos, o desempenho do aplicativo Flash/AIR é afetado pela comunicação com o servidor de licença do Adobe Access.

`DRMManager.storeVoucher()` permite que o aplicativo Flash/AIR envie vouchers de DRM que obteve fora de banda para o tempo de execução do Flash Player/AIR. O tempo de execução pode então ignorar o processo de solicitação de licença e usar os vouchers encaminhados para reproduzir os vídeos criptografados. O voucher de DRM ainda precisa ser gerado pelo servidor de licença do Adobe Access antes que possa ser obtido fora de banda. No entanto, você tem a opção de hospedar os vouchers em qualquer servidor de HTTP, em vez de um servidor de licença do Adobe Access público.

`DRMManager.storeVoucher()` também é usado para suportar o compartilhamento de voucher de DRM entre vários dispositivos. No Adobe Access 3.0, esse recurso é referido como "suporte de domínio". Caso a sua utilização suporte esse caso de uso, você poderá registrar várias máquinas em um grupo de dispositivos usando o método `DRMManager.addToDeviceGroup()`. Caso exista uma máquina com um voucher válido vinculado a um domínio, o aplicativo do AIR poderá então extrair os vouchers de DRM seriados usando o método `DRMVoucher.toByteArray()`, e nas outras máquinas você poderá importar os vouchers usando o método `DRMManager.storeVoucher()`.

Registro do dispositivo

Os vouchers de DRM são vinculados à máquina do usuário final. Consequentemente, os aplicativos Flash /AIR precisarão de um ID exclusivo para a máquina do usuário consultar o objeto voucher de DRM seriado correto. A seguinte situação retrata um processo de registro do dispositivo:

Supondo que você executou as seguintes tarefas:

- Você ativou o Adobe Access Server SDK.
- Ativou um servidor de HTTP para obter licenças geradas previamente.
- Criou um aplicativo Flash para visualizar o conteúdo protegido.

A fase de registro do dispositivo envolve as seguintes ações:

1. O aplicativo Flash cria um ID gerado aleatoriamente.
2. O aplicativo Flash executa o método `DRMManager.authenticate()`. O aplicativo deve incluir o ID gerado aleatoriamente na solicitação de autenticação. Por exemplo, inclua o ID no campo do nome do usuário.
3. A ação mencionada na etapa 2 causará o envio de uma solicitação de autenticação por parte do Adobe Access ao servidor do cliente. Essa solicitação inclui o certificado do dispositivo.
   a. O servidor extrai o certificado do dispositivo e o ID gerado da solicitação e os armazena.

   b O sub-sistema do cliente gera previamente licenças para esse certificado do dispositivo, armazena-as e permite o acesso a elas de uma maneira a associá-las ao ID gerado.

4 O servidor responde à solicitação com uma mensagem de êxito.

5 O aplicativo Flash armazena o ID gerado localmente em um Objeto Local Compartilhado (Local Shared Object - LSO).

Após o registro do dispositivo, o aplicativo Flash utiliza o ID gerado da mesma maneira que usaria o ID do dispositivo no esquema anterior:

1 O aplicativo Flash tenta localizar o ID gerado no LSO.

2 Caso o ID gerado seja encontrado, o aplicativo Flash utilizará o ID gerado enquanto baixar as licenças geradas previamente. O aplicativo Flash envia as licenças ao cliente do Adobe Access para consumo, usando o método `DRMManager.storeVoucher()`.

3 Caso o ID gerado não seja encontrado, o aplicativo Flash executará o procedimento de registro do dispositivo.

Redefinição do padrão

Quando o usuário do dispositivo executa a opção de redefinição do padrão, o certificado do dispositivo é removido. Para continuar a reproduzir o conteúdo protegido, o aplicativo Flash precisa executar novamente o procedimento de registro do dispositivo. Se o aplicativo Flash enviar uma licença gerada previamente inválida, o cliente do Adobe Access irá recusá-la, já que a licença foi criptografada para um ID de dispositivo mais antigo.

Suporte de domínio

Flash Player 11 e posterior, Adobe AIR 3.0 e posterior

Se os metadados de conteúdo especificarem que o registro do domínio é necessário, o aplicativo do AIR poderá executar uma API para conectar um grupo de dispositivos. Essa ação ativa a solicitação de registro do domínio para ser enviada ao servidor de domínio. Uma vez que uma licença seja emitida para um grupo de dispositivo, poderá ser exportada e compartilhada com outros dispositivos que foram conectados ao grupo de dispositivos.

As informações do grupo de dispositivos são então usadas no objeto `VoucherAccessInfo` do `DRMContentData`, que por sua vez será usado para apresentar as informações necessárias para recuperar e consumir um voucher com êxito.

Reprodução do conteúdo criptografado usando o suporte de domínio

Para reproduzir o conteúdo criptografado usando o Adobe Access, execute as seguintes etapas:

1 Usando `VoucherAccessInfo.deviceGroup`, verifique se o registro do grupo de dispositivo é obrigatório.

2 Caso a autenticação seja obrigatória:

   a Use a propriedade `DeviceGroupInfo.authenticationMethod` para descobrir se a autenticação é obrigatória.

   b Caso a autenticação seja obrigatória, autentique o usuário realizando UMA das seguintes etapas:

   - Obtenha o nome de usuário e a senha do usuário. Execute `DRMManager.authenticate(deviceGroup.serverURL, deviceGroup.domain, nome de usuário, senha)`.

- Obtenha um token de autenticação gerado previamente/armazenado em cache e execute `DRMManager.setAuthenticationToken()`.

c Execute `DRMManager.addToDeviceGroup()`.

3 Obtenha o voucher do conteúdo realizando uma das seguintes etapas:

a Use o método `DRMManager.loadVoucher()`.

b Obtenha o voucher de um dispositivo diferente registrado no mesmo grupo de dispositivos. Forneça o voucher para `DRMManager` por meio do método `DRMManager.storeVoucher()`.

4 Reproduza o conteúdo usando o método `NetStream.play()` .

Para exportar a licença para o conteúdo, qualquer um dos dispositivos pode fornecer os bytes brutos da licença usando o método `DRMVoucher.toByteArray()` após obter a licença do servidor de licença do Adobe Access. Os fornecedores de conteúdo geralmente limitam o número de dispositivos em um grupo de dispositivos. Caso o limite seja alcançado, você poderá precisar acionar o método `DRMManager.removeFromDeviceGroup()` em um dispositivo não usado antes de registrar o dispositivo atual.

Visualização da licença

O aplicativo Flash pode enviar uma solicitação para a visualização da licença, o que significa que o aplicativo pode executar um operação de visualização antes de pedir que o usuário compre o conteúdo a fim de determinar se a máquina do usuário possui de fato todos os critérios necessários para a reprodução. A visualização da licença refere-se à possibilidade de o cliente visualizar a licença (para ver quais direitos a licença permite), ao contrário de visualizar o conteúdo (ver uma porção pequena do conteúdo antes de decidir comprá-lo). Alguns dos parâmetros exclusivos de cada máquina são: as saídas disponíveis e seu status de proteção; a versão disponível do tempo de execução/DRM e o nível de segurança do cliente DRM. O modo de visualização da licença permite que o cliente do tempo de execução/DRM teste a lógica comercial do servidor de licença e forneça informações ao usuário para que ele possa tomar uma decisão mais informada. Dessa forma, o cliente poderá saber como uma licença válida é apresentada, mas não recebe efetivamente a chave para decodificar o conteúdo. O suporte para a visualização da licença será opcional e somente necessário se você implementar um aplicativo customizado que use essa funcionalidade.

Entrega do conteúdo

O Adobe Access não confia no mecanismo de entrega do conteúdo, uma vez que o Flash Player simplifica a camada de rede e simplesmente fornece o conteúdo protegido ao subsistema do Adobe Access. Consequentemente, o conteúdo pode ser entregue através de HTTP, HTTP Dynamic Streaming, RTMP ou RTMPE.

No entanto, você pode ter problemas devido à necessidade de metadados do conteúdo protegido (geralmente no formato de um arquivo ".metadata") antes que o Adobe Access possa adquirir uma licença para decodificar o conteúdo. Especificamente, com o protocolo RTMP/RTMPE somente dados FLV e F4V podem ser entregues ao cliente através do Flash Media Server (FMS). Por essa razão, o cliente deverá salvar o blob de metadados através de outros meios. Uma opção para solucionar esse problema é hospedar os metadados no servidor de web HTTP e implementar o reprodutor de vídeo do cliente para salvar os metadados adequados, dependendo do conteúdo a ser reproduzido no fundo.

```
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

O Open Source Media Framework (OSMF) é um sistema baseado em ActionScript que dá a você a flexibilidade e o controle completos ao criar suas próprias experiências em rich media. Para obter mais informações sobre o OSMF, visite o Site do desenvolvedor de OSMF.

Fluxo de trabalho para a reprodução de conteúdo protegido

1 Crie uma instância do `MediaPlayer`.

```
player = new MediaPlayer();
```

2 Registre o evento `MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE` no reprodutor. Esse evento será despachado caso o conteúdo seja protegido por DRM.

```
player.addEventListener(MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE,
onDRMCapabilityChange);
```

3 No manipulador de eventos, obtenha a instância `DRMTrait`. `DRMTrait` é a interface através da qual você executa os métodos relacionados ao DRM, por exemplo, `authenticate()`. Ao carregar um conteúdo protegido por DRM, o OSMF realiza as ações de validação de DRM e despacha eventos de estado. Adicione um manipulador de eventos `DRMEvent.DRM_STATE_CHANGE` a `DRMTrait`.

```
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 Manipule os eventos DRM no método `onDRMStateChange()`.

```
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;
    }
}
```

Capítulo 28: Inclusão de conteúdo em PDF no AIR

Adobe AIR 1.0 e posterior

Os aplicativos em execução no Adobe® AIR® podem produzir não só conteúdo SWF e HTML, mas também em PDF. Aplicativos AIR processam aplicativos em PDF usando a classe HTMLLoader, o mecanismo WebKit e o plug-in para navegador Adobe® Reader®. Em um aplicativo do AIR, o conteúdo em PDF pode alongar-se pela atura e largura inteiras de seu aplicativo ou, como alternativa, como uma parte da interface. O plug-in de navegador do Adobe Reader controla a exibição de arquivos PDF em um aplicativo do AIR. as modificações na interface de barra de ferramentas do Reader (como controles de posição, ancoragem e visibilidade) persistem na visualização subsequente de arquivos PDF nos aplicativos AIR e no navegador.

Importante: *Para processar conteúdo em PDF no AIR, o usuário deve ter o Adobe Reader ou Adobe® Acrobat® versão 8.1 ou posterior instalada.*

Detecção de recurso de PDF

Adobe AIR 1.0 e posterior

Se o usuário não tiver o Adobe Reader ou do Adobe Acrobat 8.1 ou posterior, o conteúdo PDF não será exibido no aplicativo do AIR. Para detectar se o usuário pode processar conteúdo em PDF, verifique primeiramente a propriedade `HTMLLoader.pdfCapability`. Essa propriedade está definida como uma das seguintes constantes da classe HTMLPDFCapability:

Constante	Descrição
HTMLPDFCapability.STATUS_OK	Uma versão suficiente (8.1 ou posterior) do Adobe Reader é detectada e o conteúdo em PDF pode ser carregado em um objeto HTMLLoader.
HTMLPDFCapability.ERROR_INSTALLED_READER_NOT_FOUND	Não foi detectada nenhuma versão do Adobe Reader. Um objeto HTMLLoader não pode exibir conteúdo em PDF.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	O Adobe Reader foi detectado, mas a versão é muito antiga. Um objeto HTMLLoader não pode exibir conteúdo em PDF.
HTMLPDFCapability.ERROR_PREFERRED_READER_TOO_OLD	Uma versão suficiente (8.1 ou posterior) do Adobe Reader foi detectada, mas a versão do aplicativo configurada para lidar com conteúdo em PDF é anterior ao Reader 8.1. Um objeto HTMLLoader não pode exibir conteúdo em PDF.

No Windows, se o Adobe Acrobat ou o Acrobat Reader versão 7.x ou posterior estiver em execução no sistema do usuário, essa versão será usada, mesmo se houver instalada uma versão posterior que ofereça suporte a PDF carregado. Nesse caso, se o valor da propriedade `pdfCapability` for `HTMLPDFCapability.STATUS_OK`, quando um aplicativo do AIR tentar carregar o conteúdo em PDF, a versão antiga do Acrobat ou do Reader exibirá um alerta (e nenhuma exceção será lançada no aplicativo do AIR). Se isso for uma possibilidade para os usuários finais, considere fornecer a eles instruções sobre como fechar o Acrobat enquanto o aplicativo é executado. Pode ser conveniente exibir essas instruções se o conteúdo em PDF não for carregado em um intervalo de tempo aceitável.

No Linux, o AIR procura o Adobe Reader no CAMINHO exportado pelo usuário (se ele contiver o comando acroread) e no diretório /opt/Adobe/Reader.

O código a seguir detecta se um usuário pode exibir conteúdo PDF em um aplicativo do AIR. Se o usuário não puder exibir um PDF, o código rastreará o código de erro correspondente ao objeto de erro HTMLPDFCapability:

```
 if(HTMLLoader.pdfCapability == HTMLPDFCapability.STATUS_OK)
{
    trace("PDF content can be displayed");
}
else
{
    trace("PDF cannot be displayed. Error code:", HTMLLoader.pdfCapability);
}
```

Carregamento de conteúdo em PDF

Adobe AIR 1.0 e posterior

Você pode adicionar um PDF a um aplicativo do AIR criando uma ocorrência HTMLLoader, definindo as dimensões e carregando o caminho de um PDF.

O exemplo a seguir carrega um PDF de um site externo. Substitua a URLRequest com o caminho para um PDF externo disponível.

```
 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);
```

Você também pode carregar conteúdo de URLs de arquivos e esquemas de URL específicos do AIR, como app e app-storage. Por exemplo, o código seguinte carrega o arquivo test.pdf no subdiretório PDFs do diretório do aplicativo:

app:/js_api_reference.pdf

Para obter mais informações sobre esquemas de URL do AIR, consulte “Esquemas de URI” na página 840.

Gravando em script o conteúdo em PDF

Adobe AIR 1.0 e posterior

Você pode usar o JavaScript para controlar o conteúdo em PDF, do mesmo modo que em um página da Web no navegador.

As extensões JavaScript para o Acrobat oferecem alguns recursos, entre eles:

- Controle de ampliação e navegação de página
- Processamento de formulários no documento
- Controle de eventos multimídia

Detalhes completos sobre extensões JavaScript para o Adobe Acrobat são fornecidos na Conexão de desenvolvedores do Adobe Acrobat, em http://www.adobe.com/devnet/acrobat/javascript.html.

Noções básicas de comunicação HTML-PDF

Adobe AIR 1.0 e posterior

O JavaScript em uma página HTML pode enviar uma mensagem para o JavaScript no conteúdo em PDF, chamando o método `postMessage()` do objeto DOM representando o conteúdo em PDF. Por exemplo, considere o seguinte conteúdo em PDF incorporado:

```
<object id="PDFObj" data="test.pdf" type="application/pdf" width="100%" height="100%"/>
```

o código JavaScript seguinte, no conteúdo HTML contido envia uma mensagem para o JavaScript no arquivo em PDF:

```
pdfObject = document.getElementById("PDFObj");
pdfObject.postMessage(["testMsg", "hello"]);
```

O arquivo em PDF pode incluir JavaScript para receber essa mensagem. Você pode adicionar o código JavaScript a arquivos em PDF em alguns contextos, incluindo os contextos em nível de documento, pasta, página, campo e lote. Apenas o contexto em nível de documento, que define scripts que são avaliados quando o documento em PDF é aberto, é discutido aqui.

Um arquivo em PDF pode adicionar uma propriedade `messageHandler` ao objeto `hostContainer`. A propriedade `messageHandler` é um objeto que define funções do manipulador para responder a mensagens. Por exemplo, o código seguinte define a função para tratar mensagens recebidas pelo arquivo em PDF a partir do contêiner host (que é o conteúdo HTML incorporando o arquivo em PDF):

```
this.hostContainer.messageHandler = {onMessage: myOnMessage};

function myOnMessage(aMessage)
{
    if(aMessage[0] == "testMsg")
    {
        app.alert("Test message: " + aMessage[1]);
    }
    else
    {
        app.alert("Error");
    }
}
```

O código JavaScript na página HTML pode chamar o método `postMessage()` do objeto PDF contido na página. Chamar esse método envia uma mensagem (`"Hello from HTML"`) para o JavaScript em nível de documento no arquivo em PDF:

```
<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>
```

Para obter um exemplo mais avançado e mais informações sobre como utilizar o Acrobat 8 para adicionar JavaScript a um arquivo PDF, consulte Cross-scripting de conteúdo PDF no Adobe AIR.

Gravação em script de conteúdo em PDF do ActionScript

Adobe AIR 1.0 e posterior

O código do ActionScript (no conteúdo SWF) não pode se comunicar diretamente com o JavaScript no conteúdo em PDF. Entretanto, o ActionScript pode se comunicar com o JavaScript na página HTML carregada em um objeto HTMLLoader que carrega conteúdo em PDF, e o código JavaScript pode se comunicar com o JavaScript no arquivo em PDF carregado. Para obter mais informações, consulte “Programação de HTML e JavaScript no AIR” na página 1014.

Limitações conhecidas do conteúdo em PDF no AIR

Adobe AIR 1.0 e posterior

O conteúdo em PDF no Adobe AIR tem as seguintes limitações:

- O conteúdo em PDF não é exibido em uma janela (um objeto NativeWindow) que seja transparente (na qual a propriedade `transparent` esteja definida como `true`).
- A ordem de exibição de um arquivo em PDF funciona de forma diferente de outros objetos de exibição em um aplicativo do AIR. Embora o conteúdo em PDF se afixe corretamente de acordo com a ordem de exibição de HTML, ele sempre ficará por cima do conteúdo na ordem de exibição do aplicativo do AIR.

- Se determinadas propriedades visuais de um objeto HTMLLoader que contenha um documento PDF forem alteradas, o documento PDF se tornará invisível. Essas propriedades incluem `filters`, `alpha`, `rotation` e `scaling`. Alterar essas propriedades torna o conteúdo em PDF invisível até a redefinição das propriedades. O conteúdo em PDF também fica invisível se você alterar essas propriedades de contêineres de objeto de exibição que contêm o objeto HTMLLoader.
- O conteúdo PDF fica visível somente quando a propriedade `scaleMode` do objeto Stage do objeto NativeWindow contendo o conteúdo PDF é definida como `StageScaleMode.NO_SCALE`. Quando ela é definida com outro valor, o conteúdo PDF não fica visível.
- Clicar em links para conteúdo dentro do arquivo em PDF atualiza a posição de rolagem do conteúdo em PDF. Clicar em links para conteúdo fora do arquivo em PDF redireciona o objeto HTMLLoader que contém o PDF (mesmo que o destino de um link seja uma nova janela).
- Fluxos de trabalho de comentário de PDF não funcionam no AIR.

Capítulo 29: Princípios básicos da interação do usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Seu aplicativo pode criar a interatividade usando o ActionScript 3.0 para responder à atividade do usuário. Essa seção supõe que você já está familiarizado com o modelo de eventos do ActionScript 3.0. Para obter mais informações, consulte "Manipulação de eventos" na página 126.

Captura da entrada do usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A interação do usuário, por meio do teclado, do mouse, da câmera ou de uma combinação desses dispositivos, é a base da interatividade. No ActionScript 3.0, identificar e responder à interação do usuário envolve principalmente ouvir eventos.

A classe InteractiveObject, uma subclasse de DisplayObject, fornece a estrutura comum dos eventos e a funcionalidade necessária para manipular a interação do usuário. Você não cria diretamente uma ocorrência da classe InteractiveObject. Em vez disso, objetos de exibição como SimpleButton, Sprite, TextField e vários componentes da ferramenta de autoria do Flash e do Flex herdam o modelo de interação do usuário dessa classe e, desse modo, compartilham uma estrutura comum. Isso significa que as técnicas aprendidas e o código que você grava para manipular a interação do usuário em um objeto derivado de InteractiveObject são aplicáveis a todos os outros usuários.

Conceitos e termos importantes

É importante que você se familiarize com os seguintes termos-chave de interação do usuário antes de continuar:

Código de caractere Um código numérico que representa um caractere no conjunto de caracteres atual (associado a uma tecla que está sendo pressionada no teclado). Por exemplo, "D" e "d" têm códigos de caractere diferentes embora tenham sido criados pela mesma tecla do teclado do alfabeto inglês.

Menu de contexto Menu que aparece quando um usuário clica com o botão direito do mouse ou usa uma combinação específica do teclado-mouse. Os comandos do menu de contexto em geral são aplicados diretamente no que foi clicado. Por exemplo, um menu de contexto de uma imagem pode conter um comando para mostrar a imagem em uma janela separada e um comando para fazer download dessa imagem.

Foco Indicação de que um elemento selecionado está ativo e é o destino da interação do teclado ou do mouse.

Código de tecla Código numérico que corresponde a uma tecla física no teclado.

Mais tópicos da Ajuda

InteractiveObject

Teclado

Mouse

ContextMenu

Gerenciamento do foco

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um objeto interativo pode receber o foco, programaticamente ou por meio de uma ação do usuário. Além disso, se a propriedade `tabEnabled` estiver definida como `true`, o usuário poderá passar o foco de um objeto para outro pressionando a tecla Tab. Observe que o valor de `tabEnabled` é `false` por padrão, exceto nos seguintes casos:

- Para um objeto SimpleButton, o valor é `true`.
- Para um campo de texto de entrada, o valor é `true`.
- Para um objeto Sprite ou MovieClip com `buttonMode` definido como `true`, o valor é `true`.

Em cada uma dessas situações, você pode adicionar um ouvinte para `FocusEvent.FOCUS_IN` ou `FocusEvent.FOCUS_OUT` a fim de fornecer outros comportamentos quando o foco for alterado. Isso é útil especialmente para campos de texto e formulários, mas também pode ser usado em entidades gráficas, clipes de filme ou qualquer objeto herdado da classe InteractiveObject. O exemplo a seguir mostra como ativar o ciclo do foco com a tecla Tab e como responder ao evento de foco subsequente. Nesse caso, cada quadrado muda de cor assim que recebe o foco.

Nota: *O Flash Professional usa atalhos do teclado para gerenciar o foco; desse modo, para simular o gerenciamento do foco corretamente, os arquivos SWF devem ser testados em um navegador ou AIR, não no Flash.*

```
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);
}
```

Descobrindo tipos de entrada

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

As versões do Flash Player 10.1 e do Adobe AIR 2 introduziram a habilidade de testar o ambiente de tempo de execução em relação ao suporte de tipos de entrada específicos. Você pode utilizar ActionScript para testar se o dispositivo no qual o tempo de execução está implantado:

- Suporta entrada de por meio de stylus ou de dedo (ou nenhuma entrada de toque).
- Tem um teclado virtual ou físico para o usuário (ou nenhum teclado).
- Mostra um cursor (caso contrário, recursos que dependem do posicionamento do cursor sobre um objeto não funcionarão).

As APIs ActionScript de descoberta de entradas incluem:

- flash.system.Capabilities.touchscreenType property: Um valor fornecido em tempo de execução indica que o tipo de entrada posui suporte no ambiente atual.
- classe flash.system.TouchScreenType: Uma classe de constantes de valor de enumeração para a propriedade Capabilities.touchscreenType.
- propriedade flash.ui.Mouse.supportsCursor: Um valor fornecido no tempo de execução indicando se um cursor persistente está disponível ou não.
- propriedade flash.ui.Keyboard.physicalKeyboardType: Um valor fornecido no tempo de execução indicando se um teclado completo físico está disponível ou apenas um teclado numérico ou nenhum teclado.
- classe flash.ui.KeyboardType : Uma classe de constantes de valor de enumeração para a propriedade flash.ui.Keyboard.physicalKeyboardType.
- propriedade flash.ui.Keyboard.hasVirtualKeyboard: Um valor fornecido no tempo de execução indicando se um teclado virtual é fornecido ao usuário (no lugar de um teclado físico ou em adição a um teclado físico).

As APIs de descoberta de entradas permitem que você aproveite das capacidades do dispositivo do usuário ou fornecem alternativas quando essas capacidades não estejam disponíveis. Estas APIs são especialmente uteis para o desenvolvimento de aplicativos móveis e sensíveis ao toque. Por exemplo, se houver uma interface para um dispositivo móvel com botões pequenos para uso de stylus, você pode fornecer uma interface alternativa com botões maiores para um usuário que utiliza toques dos dedos como entrada. O seguinte código se refere a um aplicativo que possui uma função createStylusUI() que atribui um conjunto de elementos de interface do usuário apropriados para interação com stylus. Outra função, chamada de createTouchUI(), atribui outro conjunto de elementos de interface do usuário apropriados para interação com dedos:

```
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();
}
```

Quando estiver desenvolvendo aplicativos para ambientes de entradas diferentes, considere o seguinte gráfico:

Ambiente	supportsCursor	touchscreenType == FINGER	touchscreenType == STYLUS	touchscreenType == NONE
Desktop tradicional	true	false	false	true
Dispositivos de tela sensível ao toque capacitivos (tablets, PDAs e telefones que detectam toques sutis, como o iPhone da Apple ou o Palm Pre)	false	true	false	false
Dispositivos de tela sensível ao toque resistivos (tablets, PDAs e telefones que detectam contatos de alta pressão precisos, como o HTC Fuze)	false	false	true	false
Dispositivos que não tem tela sensível ao toque (telefones básicos e dispositivos que executam aplicativos, mas não possuem telas que detectem contato)	false	false	false	true

Nota: *Plataformas de dispositivos diferentes podem suportar várias combinações de tipos de entradas. Utilize este gráfico como um guia geral.*

Capítulo 30: Entrada do teclado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Seu aplicativo pode capturar e responder à entrada do teclado e pode manipular um IME para permitir que os usuários digitem caracteres em texto não ASCII em idiomas de múltiplos bytes. Essa seção supõe que você já está familiarizado com o modelo de eventos do ActionScript 3.0. Para obter mais informações, consulte "Manipulação de eventos" na página 126.

Para obter informações sobre como descobrir que tipo de suporte à teclado está disponível (como físico, virtual alfanumérico ou numérico de 12 botões) durante o tempo de execução, consulte "Descobrindo tipos de entrada" na página 575.

Um Editor de método de entrada (Input Method Editor (IME)) permite que os usuários digitem caracteres complexos e símbolos, utilizando um teclado padrão. Você pode utilizar classes IME para permitir que os usuários usufruam das vantagens do IME do sistema em seus aplicativos.

Mais tópicos da Ajuda

flash.events.KeyboardEvent

flash.system.IME

Captura da entrada do teclado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os objetos de exibição que herdam o modelo de interação da classe InteractiveObject podem responder aos eventos de teclado usando ouvintes de eventos. Você pode colocar, por exemplo, um ouvinte de eventos no palco para ouvir e responder à entrada do teclado. No exemplo a seguir, um ouvinte de eventos captura um pressionamento de tecla e as propriedades de código e nome da tecla são exibidos:

```
function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (character code: " +
event.charCode + ")");
}
stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Algumas teclas, como a tecla Ctrl, geram eventos embora não tenha nenhuma representação de glifo.

No exemplo de código anterior, o ouvinte de eventos de teclado capturou a entrada do teclado para todo o palco. Você também pode gravar um ouvinte de eventos para um objeto de exibição específico no palco; esse ouvinte de eventos é acionado quando o objeto está em foco.

No exemplo a seguir, os pressionamentos de tecla são refletidos no painel Saída somente quando o usuário digita dentro da ocorrência TextField. Manter a tecla Shift temporariamente pressionada faz com que a cor da borda de TextField mude para vermelho.

Esse código supõe a existência de uma ocorrência TextField chamada `tf` no palco.

```
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;
    }
}
```

A classe TextField também registra um evento `textInput` que pode ser ouvido quando um usuário insere algum texto. Para obter mais informações, consulte "Captura de entrada de texto" na página 389.

Nota: *No tempo de execução do AIR, um evento de teclado pode ser cancelado. No tempo de execução do Flash Player, o evento de teclado não pode ser cancelado.*

Códigos de teclas e de códigos de caracteres

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode acessar as propriedades `keyCode` e `charCode` de um evento de teclado para determinar qual tecla foi pressionada e, em seguida, acionar outras ações. A propriedade `keyCode` é um valor numérico que corresponde ao valor de uma tecla no teclado. A propriedade `charCode` é o valor numérico dessa tecla no conjunto de caracteres atual. O conjunto de caracteres padrão é UTF-8, que oferece suporte para ASCII.

A principal diferença entre o código de tecla e os valores de caractere está no fato de o código de tecla representar uma tecla específica do teclado (o valor 1 de um teclado é diferente do valor 1 na linha superior, mas a tecla que gera "1" e a que gera "!" são iguais), enquanto o valor de caractere representa um caractere específico (os caracteres R e r são diferentes).

Nota: *Consulte as correlações entre as teclas e seus valores de códigos de caracteres em ASCII na classe flash.ui.Keyboard da Referência do ActionScript® 3.0 para Adobe® Flash® Platform.*

O mapeamento entre as teclas e os códigos de tecla dependem do dispositivo e do sistema operacional. Devido a isso, não use mapeamentos de tecla para acionar comandos. Em vez disso, use os valores de constantes predefinidos fornecidos pela classe Keyboard para fazer referência às propriedades `keyCode` adequadas. Por exemplo, em vez de usar o mapeamento para a tecla Shift, use a constante `Keyboard.SHIFT` (como mostra o código de exemplo anterior).

Precedência de KeyboardEvent

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Assim como outros eventos, a sequência de eventos de teclado é determinada pela hierarquia de objetos de exibição, não pela ordem em que os métodos `addEventListener()` são atribuídos no código.

Suponha, por exemplo, que você coloque um campo de texto chamado `tf` em um clipe de filme chamado `container` e adicione um ouvinte de eventos para um evento de teclado nas duas ocorrências, como mostra o exemplo a seguir:

```
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 + ")");
}
```

Como há um ouvinte no campo de texto e no recipiente pai, a função `reportKeyDown()` é chamada duas vezes para cada pressionamento de tecla em TextField. Para cada tecla pressionada, o campo de texto envia um evento antes que o clipe de filme `container` faça isso.

O sistema operacional e o navegador da Web processam eventos de teclado antes do Adobe Flash Player ou AIR. No Microsoft Internet Explorer, por exemplo, pressionar Ctrl+W fecha a janela do navegador antes que qualquer arquivo SWF contido envie um evento de teclado.

Uso da classe IME

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe IME permite manipular o IME do sistema operacional de dentro do Flash Player ou do Adobe AIR.

Usando o ActionScript, é possível determinar o seguinte:

- Se um IME está instalado no computador do usuário (`Capabilities.hasIME`)
- Se o IME está ativado ou desativado no computador do usuário (`IME.enabled`)
- O modo de conversão do IME atual está usando (`IME.conversionMode`)

É possível associar um campo de texto de entrada a um contexto de IME específico. Ao alternar entre campos de texto, também é possível alternar o IME entre Hiragana (japonês), números de largura total, números de meia largura, entrada direta e assim por diante.

Um IME permite que os usuários digitem caracteres de texto não ASCII em idiomas com vários bytes, como chinês, japonês e coreano.

Para obter mais informações sobre como trabalhar com IMEs, consulte a documentação do sistema operacional no qual você está desenvolvendo o aplicativo. Para obter recursos adicionais, consulte os seguintes sites:

- http://www.msdn.microsoft.com/goglobal/
- http://developer.apple.com/library/mac/navigation/
- http://www.java.sun.com/

Nota: *Se o IME não estiver ativo no computador do usuário, chamadas para métodos ou propriedades do IME, que não sejam* `Capabilities.hasIME`*, falharão. Depois de ativar manualmente o IME, chamadas subsequentes do ActionScript para métodos e propriedades do IME funcionarão conforme o esperado. Por exemplo, se estiver usando um IME japonês, você deverá ativá-lo para poder chamar qualquer método ou propriedade do IME.*

Verificação da instalação e ativação do IME

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Antes de chamar qualquer um dos métodos ou propriedades do IME, você deve sempre verificar se o computador do usuário tem um IME instalado e ativado. O código a seguir ilustra como verificar se o usuário tem um IME instalado e ativo antes de chamar qualquer método:

```
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.");
}
```

O código anterior primeiro verifica se o usuário tem um IME instalado usando a propriedade `Capabilities.hasIME`. Se essa propriedade estiver definida como `true`, o código verificará se o IME do usuário está ativado usando a propriedade `IME.enabled`.

Determinação de qual modo de conversão do IME está ativado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao criar aplicativos multilíngue, talvez seja necessário determinar qual modo de conversão está ativo para o usuário. O código a seguir demonstra como verificar se o usuário tem um IME instalado e, nesse caso, qual modo de conversão do IME está ativo no momento:

```
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.";
}
```

O código anterior primeiro verifica se o usuário tem um IME instalado. Em seguida, ele verifica qual modo de conversão o IME atual está usando, comparando a propriedade `IME.conversionMode` com cada uma das constantes na classe IMEConversionMode.

Configuração do modo de conversão do IME

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando você altera o modo de conversão do IME do usuário, é necessário verificar se o código está inserido em um bloco `try..catch`, porque a configuração de um modo de conversão usando a propriedade `conversionMode` poderá emitir um erro, se o IME não puder definir o modo de conversão. O código a seguir demonstra como usar um bloco `try..catch` ao definir a propriedade `IME.conversionMode`:

```
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;
    }
}
```

O código anterior primeiro cria um campo de texto que é usado para exibir uma mensagem de status para o usuário. Em seguida, se o IME estiver instalado, o código ativará o IME e definirá o modo de conversão como coreano. Se o computador do usuário não tiver um IME coreano instalado, o Flash Player ou o AIR emitirá um erro que será capturado pelo bloco `try..catch`. O bloco `try..catch` exibe a mensagem de erro no campo de texto criado anteriormente.

Desativação do IME para determinados campos de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Em alguns casos, convém desativar o IME do usuário enquanto ele digita caracteres. Por exemplo, se você tiver um campo de texto que aceita apenas entrada numérica, é conveniente ativar o IME e diminuir a velocidade da entrada de dados.

O exemplo a seguir demonstra como é possível ouvir os eventos `FocusEvent.FOCUS_IN` e `FocusEvent.FOCUS_OUT` e desativar o IME do usuário de maneira correspondente:

```
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;
    }
}
```

Esse exemplo cria dois campos de texto de entrada, `phoneTxt` e `nameTxt` e, em seguida, adiciona dois ouvintes de eventos ao campo de texto `phoneTxt`. Quando o usuário define o foco para o campo de texto `phoneTxt`um evento `FocusEvent.FOCUS_IN` é despachado e o IME é desativado. Quando o campo de texto `phoneTxt` perde o foco, o evento `FocusEvent.FOCUS_OUT` é despachado para reativar o IME.

Ouvir eventos de composição do IME

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os eventos de composição do IME são despachados quando uma string de composição está sendo definida. Por exemplo, se o usuário tiver o IME habilitado e ativo e digitar uma string em japonês, o evento `IMEEvent.IME_COMPOSITION` despachará assim que o usuário selecionar a string da composição. Para ouvir o evento `IMEEvent.IME_COMPOSITION`, é necessário adicionar um ouvinte de eventos à propriedade estática `ime` na classe System (`flash.system.System.ime.addEventListener(...)`), conforme mostrado no exemplo a seguir:

```
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;
}
```

O código anterior cria dois campos de texto e os adiciona à lista de exibição. O primeiro campo de texto, `inputTxt`, é um campo de texto de entrada que permite que o usuário digite texto japonês. O segundo campo de texto, `outputTxt`, é um campo de texto dinâmico que exibe mensagens de erro para o usuário ou ecoa a string em japonês que o usuário digita no campo de texto `inputTxt`.

Teclados virtuais

Flash Player 10.2 e posterior, AIR 2.6 e posterior

Dispositivos móveis tais como telefones e tablets frequentemente fornecem um software de teclado virtual em vez de um teclado físico. As classes da API Flash permitem:

- Detectar quando o teclado virtual é exibido e quando é fechado.
- Impedir a exibição do teclado virtual.

- Determinar a área do palco obscurecida pelo teclado virtual.
- Criar objetos interativos que ativam o teclado virtual quando estão em foco. (Não suportado pelos aplicativos do AIR que rodam no iOS.)
- (Somente para o AIR) Desativa o comportamento de panorama automático para que o seu aplicativo possa modificar sua própria exibição a fim de se adaptar ao teclado.

Controle do comportamento do teclado virtual

O tempo de execução abre automaticamente o teclado virtual quando o usuário toca dentro de um campo de texto ou um objeto interativo especialmente configurado. Quando o teclado é aberto, o tempo de execução segue as convenções de plataforma nativa deslocando e redimensionando o conteúdo do aplicativo para que o usuário possa ver o texto enquanto digita.

Quando o teclado é aberto, o objeto focado envia os seguintes eventos em sequência:

Evento `softKeyboardActivating` — enviado imediatamente antes do teclado começar a aparecer na plataforma. Se você chamar o método `preventDefault()` do objeto do evento enviado, o teclado virtual não é aberto.

`softKeyboardActivate` evento — enviado após a manipulação do evento `softKeyboardActivating` terminar. Quando o objeto focado envia esse evento, a propriedade`softKeyboardRect` do objeto Stage foi atualizado para refletir a área da plataforma obscurecida pelo teclado virtual. Este evento não pode ser cancelado.

Nota: *Se o teclado muda de tamanho, quando o usuário muda o tipo de teclado, por exemplo, o objeto focado envia um segundo evento softKeyboardActivate.*

Evento `softKeyboardDeactivate` — enviado quando o teclado virtual é fechado por algum motivo. Este evento não pode ser cancelado.

O exemplo a seguir adiciona dois objetos TextField na plataforma. O TextField superior impede o teclado de surgir quando você toca o campo e fecha-o se já estiver aparecendo. O TextField inferior demonstra o comportamento padrão. O exemplo informa os eventos de teclado de software enviados por ambos os campos de texto.

```
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 );
    }
}
}
```

Adicionando suporte a teclado virtual para objetos interativos

Flash Player 10.2 e superior, AIR 2.6 e superior (mas não suportado no iOS)

Geralmente o teclado virtual só é aberto quando um objeto TextField é tocado. Você pode configurar uma instância da classe InteractiveObject para abrir o teclado virtual quando esse está em foco.

Para configurar uma instância do InteractiveObject para abrir o teclado virtual, defina sua propriedade `needsSoftKeyboard` como `true`. Sempre que o objeto é atribuído à propriedade focus do palco, o teclado virtual abre-se automaticamente. Além disso, você pode ativar o teclado virtual chamando o método `requestSoftKeyboard()` do InteractiveObject.

O exemplo seguinte ilustra como você pode programar um InteractiveObject para atuar como um campo de entrada de texto. A classe TextInput exibida no exemplo define a propriedade `needsSoftKeyboard` para que o teclado seja ativado quando necessário. O objeto escuta então os eventos `keyDown` e insere o caractere digitado no campo.

O exemplo usa o mecanismo de texto do Flash para acrescentar e exibir qualquer texto digitado e manipula alguns eventos importantes. Para simplificar, o exemplo não implementa um campo de texto com recursos completos.

```
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();
        }
    }
}
```

A seguinte classe de aplicação principal ilustra como usar a classe TextInput e gerenciar o layout do aplicativo quando o teclado estiver ativado ou quando a orientação do dispositivo for alterada. A classe main cria um objeto TextInput e define seus limites para preencher o palco. A classe ajusta o tamanho do objeto TextInput quando o teclado virtual é ativado ou quando o palco muda de tamanho. A classe escuta os eventos do teclado virtual a partir do objeto TextInput e redimensiona os eventos a partir do palco. Independentemente da causa do evento, o aplicativo determina a área visível do palco e redimensiona o controle de entrada para preenchê-lo. Naturalmente, em uma aplicação real, você precisaria de um algoritmo de layout mais sofisticado.

```
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;
        }
    }
}
```

Nota: *O palco despacha somente eventos de redimensionamento em resposta a uma alteração de orientação quando a propriedade* `scaleMode` *é definida como* `noScale`*. Em outros modos, as dimensões do palco não se alteram; em vez disso, o conteúdo é escalado para compensar.*

Tratamento de mudanças de exibição do aplicativo

AIR 2.6 e posterior

No AIR, você pode desativar o panorama padrão e redimensionar o comportamento associado à ativação do teclado virtual, definindo o elemento `softKeyboardBehavior`, no descritor do aplicativo, como `none`:

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Quando você desativar o comportamento automático, é responsabilidade do seu aplicativo fazer quaisquer ajustes necessários para a exibição do aplicativo. Um evento softKeyboardActivate é despachado quando o teclado é aberto. No momento em que o evento `softKeyboardActivate` é enviado, a propriedade `softKeyboardRect` do palco contém as dimensões da área obscurecida pelo teclado aberto. Use essas dimensões para mover ou redimensionar seu conteúdo, de maneira que esse seja exibido adequadamente enquanto o teclado é aberto e o usuário digita. (Quando o teclado está fechado, as dimensões do retângulo softKeyboardRect são todas zero.)

Quando o teclado é fechado, um evento `softKeyboardDeactivate` é enviado, e é possível voltar a exibição do aplicativo ao normal.

```
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;
        }
    }
}
```

Nota: *No Android, existem circunstâncias, incluindo o modo tela cheia, nas quais as dimensões exatas do teclado não estão disponíveis no sistema operacional. Nesses casos, o tamanho é estimado. Do mesmo modo, nas orientações de paisagem, o teclado IME de tela cheia nativo é usado para todas as entradas de texto. Esse teclado IME possui um campo de entrada de texto incorporado e oculta o palco inteiro. Não existe maneira de ativar um teclado com orientação de paisagem sem preencher a tela.*

Capítulo 31: Entrada do mouse

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Seu aplicativo pode criar interatividade capturando e respondendo à entrada do mouse. Essa seção supõe que você já está familiarizado com o modelo de eventos do ActionScript 3.0. Para obter mais informações, consulte "Manipulação de eventos" na página 126.

Para obter informações sobre como descobrir que tipo de suporte à mouse está disponível (como cursor persistente , stylus ou entrada de toques) durante o tempo de execução, consulte "Descobrindo tipos de entrada" na página 575.

Mais tópicos da Ajuda

flash.ui.Mouse

flash.events.MouseEvent

"Toque, multitoque e entrada de gestos" na página 599

Captura da entrada do mouse

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os cliques criam eventos de mouse que podem ser usados para acionar funções interativas. Você pode adicionar um ouvinte de eventos ao palco para ouvir eventos de mouse que ocorrem em qualquer lugar no arquivo SWF. Também é possível adicionar ouvintes de eventos a objetos no palco que herdam modelos de InteractiveObject (por exemplo, Sprite ou MovieClip); esses ouvintes são acionados quando o objeto é clicado.

Assim como os eventos de teclado, os eventos de mouse aparecem em balões. No exemplo a seguir, como `square` é filho de Stage, o evento é enviado a partir do objeto `square` da entidade gráfica, bem como do objeto Stage quando se clica em square:

```
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 + "]");
}
```

No exemplo anterior, observe que o evento de mouse contém informações de posição sobre o clique. As propriedades `localX` e `localY` contêm o local do clique no filho mais inferior da cadeia de exibição. Por exemplo, clicar no canto superior esquerdo de `square` registra as coordenadas de local [0,0] porque este é o ponto de registro de `square`. Como alternativo, as propriedades `stageX` e `stageY` referem-se às coordenadas globais do clique no palco. O mesmo clique registra [50,50] para essas coordenadas, pois `square` foi movido até elas. Esses dois pares de coordenadas podem ser úteis dependendo do modo como deseja responder à interação do usuário.

Nota: *No modo de tela cheia, você pode configurar o aplicativo para usar o travamento de mouse. O travamento do mouse desativa o cursor e permite o movimento sem restrições do mouse. Para obter mais informações, consulte* *"Trabalho com o modo de tela cheia" na página 169.*

O objeto MouseEvent também contém as propriedades boolianas `altKey`, `ctrlKey` e `shiftKey`. Você pode usar essas propriedades para verificar se a tecla Alt, Ctrl ou Shift também está sendo pressionada ao mesmo tempo em que o clique do mouse.

Arrastando Sprites pelo palco

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode permitir que os usuários arrastem um objeto Sprite pelo palco usando o método `startDrag()` da classe Sprite. O código a seguir mostra um exemplo disso:

```
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);
}
```

Para obter mais detalhes, consulte a seção sobre a criação da interação de movimentação do mouse em "Alteração da posição" na página 176.

Arrastar e soltar no AIR

No Adobe AIR, você pode permitir a ação de arrastar e soltar para que os usuários arrastem dados para o seu aplicativo e a partir dele. Consulte mais detalhes em "Arrastar e soltar no AIR" na página 627.

Personalização do cursor do mouse

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O cursor do mouse (ponteiro) pode ser ocultado ou alternado para qualquer objeto de exibição no palco. Para ocultar o cursor do mouse, chame o método `Mouse.hide()`. Personalize o cursor chamando `Mouse.hide()`, ouvindo o evento `MouseEvent.MOUSE_MOVE` no palco e definindo as coordenadas de um objeto de exibição (seu cursor personalizado) com as propriedades `stageX` e `stageY` do evento. O exemplo a seguir ilustra uma execução básica dessa tarefa:

```
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;
}
```

Exemplo de entrada de mouse: WordSearch

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Este exemplo demonstra a interação do usuário por meio da manipulação de eventos de mouse. Os usuários criam o máximo de palavras possível a partir de uma grade aleatória de letras, movendo-se na horizontal ou vertical na grade, mas nunca usando a mesma letra duas vezes. Este exemplo demonstra os seguintes recursos do ActionScript 3.0:

- Criação dinâmica de uma grade de componentes
- Resposta a eventos de mouse
- Manutenção da pontuação com base na interação do usuário

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo WordSearch estão localizados na pasta Amostras/WordSearch. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
WordSearch.as	A classe que fornece a funcionalidade principal do aplicativo.
WordSearch.fla ou WordSearch.mxml	O arquivo principal do aplicativo para Flex (MXML) ou Flash (FLA).
dictionary.txt	Um arquivo usado para determinar se as palavras criadas podem ser pontuadas e foram escritas corretamente.

Carregamento de um dicionário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para criar um jogo que envolve a busca de palavras, um dicionário é necessário. O exemplo inclui um arquivo de texto chamado dictionary.txt que contém uma lista de palavras separadas por retornos de carro. Depois que uma matriz chamada `words` é criada, a função `loadDictionary()` solicita esse arquivo que, ao ser carregado com êxito, se transforma em uma longa string. É possível analisar essa string em uma matriz de palavras usando o método `split()`, quebrando-a em cada ocorrência de um retorno de carro (código de caractere 10) ou de uma nova linha (código de caractere 13). Essa análise ocorre na função `dictionaryLoaded()`:

```
words = dictionaryText.split(String.fromCharCode(13, 10));
```

Criação da interface de usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Depois de armazenar as palavras, você pode configurar a interface de usuário. Crie duas ocorrências de botão: uma para enviar uma palavra e outra para apagar uma palavra que esteja com erros ortográficos no momento. Em cada caso, responda à entrada do usuário ouvindo o evento `MouseEvent.CLICK` que o botão transmite e, em seguida, chamando uma função. Na função `setupUI()`, esse código cria os ouvintes nos dois botões:

```
submitWordButton.addEventListener(MouseEvent.CLICK,submitWord);
clearWordButton.addEventListener(MouseEvent.CLICK,clearWord);
```

Geração de uma placa de jogo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A placa de jogo é uma grade de letras aleatórias. Na função `generateBoard()`, uma grade bidimensional é criada, aninhando um loop em outro. O primeiro loop aumenta as linhas e o segundo aumenta o número total de colunas por linha. Cada célula criada por essas linhas e colunas contém um botão que representa uma letra na placa.

```
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);
        }
    }
}
```

Embora um ouvinte seja adicionado para um evento `MouseEvent.CLICK` apenas em uma linha, por estar em um loop `for`, ele é atribuído a cada ocorrência de botão. Além disso, cada botão recebe um nome derivado de sua posição de linha e coluna, o que facilita a referência à linha e à coluna de cada botão posteriormente no código.

Criação de palavras a partir da entrada do usuário

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para criar palavras, é possível selecionar letras que estão próximas na vertical ou na horizontal, mas nunca se deve usar a mesma letra duas vezes. Cada clique gera um evento de mouse e, nesse momento, a palavra fornecida pelo usuário deve ser verificada para assegurar que esteja sendo corretamente formada a partir das letras clicadas antes. Em caso negativo, a palavra anterior é removida e uma nova é iniciada. Essa verificação ocorre no método `isLegalContinuation()`.

```
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));
}
```

Os métodos `charAt()` e `indexOf()` da classe String recuperam as linhas e colunas apropriadas do botão clicado atualmente e do botão clicado anteriormente. O método `isLegalContinuation()` retornará `true` se a linha ou coluna estiver inalterada e se a linha ou coluna que foi alterada estiver em um único incremento da anterior. Se desejar alterar as regras do jogo e permitir a formação de palavras na diagonal, remova as verificações de linhas ou colunas inalteradas; a linha final é similar a:

```
return (Math.abs(prevButtonRow - currButtonRow) <= 1) && Math.abs(prevButtonCol -
currButtonCol) <= 1));
```

Verificação do envio de palavras

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para concluir o código do jogo, os mecanismos de verificação do envio de palavras e de cálculo da pontuação são necessários. O método `searchForWord()` contém os dois:

```
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;
}
```

Essa função percorre todas as palavras do dicionário. Se a palavra do usuário coincidir com a do dicionário, sua posição no dicionário será retornada. O método `submitWord()` verifica a resposta e atualiza a pontuação se a posição for válida.

Personalização

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No início da classe, existem várias constantes. Você pode modificar esse jogo modificando essas variáveis. É possível, por exemplo, alterar a quantidade de tempo disponível para jogar aumentando a variável `TOTAL_TIME`. Você também pode aumentar ligeiramente a variável `PERCENT_VOWELS` para aumentar a probabilidade de busca das palavras.

Capítulo 32: Toque, multitoque e entrada de gestos

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

Os recursos de manuseio de evento de toque da plataforma Flash incluem a entrada de um único ponto de contato ou múltiplos pontos de contato em dispositivos sensíveis ao toque. Além disso, o tempo de execução do Flash trata de eventos que combinam múltiplos pontos de toque com o movimento criado pelo gesto. Em outras palavras, os tempos de execução do Flash interpretam dois tipos de entrada:

Toque entrada com um dispositivo de ponto único como, por exemplo, um dedo, uma stylus ou outra ferramenta em um dispositivo sensível ao toque. Alguns dispositivos possuem suporte a pontos de contato simultâneos com um dedo ou uma stylus.

Multitoque entrada com mais de um ponto simultâneo de contato.

Gesto Entrada interpretada por um dispositivo ou sistema operacional em resposta a um ou mais eventos de toque. Por exemple, o usuário gira dois dedos simultaneamente e o dispositivo ou sistema operacional interpreta o toque como um gesto de rotação. Alguns gestos são executados com um dedo ou ponto de toque e, outros gestos podem necessitar de múltiplos pontos de toque. O dispositivo ou o sistema operacional estabelece o tipo de gesto a ser atribuído à entrada.

Ambas entradas de toque e gesto podem ser de entrada multitoque, dependendo do dispositivo do usuário. O ActionScript fornece uma API para tratar de eventos de toque, de gestos e eventos rastreados individualmente para entrada multitoque.

Nota: *A escuta de eventos de toque e gesto pode consumir uma quantidade significativa de recursos (equivalente à renderização de vários quadros por segundo), dependendo do dispositivo de computação e do sistema operacional. Geralmente é melhor usar eventos de mouse quando você não precisa efetivamente da funcionalidade extra proporcionada por toques ou gestos. Ao usar eventos de toque ou gesto, considere reduzir a quantidade de alterações gráficas que podem ocorrer, especialmente quando tais eventos podem ser despachados rapidamente, seja durante uma pan, seja durante uma operação de rotação ou zoom. Por exemplo, você poderia parar a animação em um componente enquanto o usuário estivesse a redimensionando com um gesto de zoom.*

Mais tópicos da Ajuda

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 (Eventos de Toque e Gesto em Dispositivos Móveis)

Mike Jones: Controladores de Jogos Virtuais

Introdução à entrada de toque

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

Quando a plataforma Flash está sendo executada em um ambiente que possui suporte a entrada de toque, as instâncias InteractiveObject podem monitorar eventos de toque e manipuladores de chamada. Geralmente, é possível manipular eventos de toque, multitoque e gestos da mesma forma que outros eventos no ActionScript (consulte "Manipulação de eventos" na página 126 para obter as informações básicas sobre como gerenciar eventos com o ActionScript).

No entanto, para que o tempo de execução Flash interprete um toque ou um gesto, o tempo de execução deve estar sendo executado em um ambiente de hardware e software que tenha suporte a entrada de toque ou multitoque. Consulte "Descobrindo tipos de entrada" na página 575 para obter um gráfico comparativo de diferentes tipos de telas sensíveis ao toque. Além disso, se o tempo de execução estiver em um aplicativo contêiner (por exemplo, um navegador), o contêiner envia a entrada para o tempo de execução. Em alguns casos, o hardware atual e o ambiente do sistema operacional possuem suporte a multitoque, mas o navegador que contém o tempo de execução do Flash interpreta a entrada e não a envia para o tempo de execução. De forma alternativa, ele pode simplesmente ignorar a entrada por completo.

O diagrama a seguir demonstra o fluxo de entrada do usuário para o tempo de execução:

Fluxo de entrada do usuário para o tempo de execução da plataforma Flash

Felizmente, a ActionScript API para desenvolvimento de aplicativos de toque inclui classes, métodos e propriedades para determinar o suporte para entrada toque e multitoque no ambiente do tempo de execução. A API utilizada para determinar o suporte à entrada de toque é a "API de descoberta" para manipulação de eventos de toque.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes para programar aplicativos de manipulação de eventos de toque:

API de descoberta Métodos e propriedades utilizados para detectar no ambiente de tempo de execução o suporte a eventos de toque e diferentes modos de entrada.

Evento de toque Uma ação de entrada executada em um dispositivo sensível ao toque utilizando um único ponto de contato.

Ponto de toque Ponto de contato de um único evento de toque. Mesmo que um dispositivo não tenha suporte à entrada de gestos, ele pode ter suporte a múltiplos pontos de toque simultâneos.

Sequência de toque Série de eventos que representam a duração de um único toque. Esses eventos incluem movimentos iniciais, zero ou mais movimentos e um movimento final.

Evento multitoque Ação de entrada feita em um dispositivo sensível ao toque utilizando diversos pontos de contato (por exemplo, mais de um dedo).

Evento de gesto Ação de entrada, executada em um dispositivo sensível ao toque que segue um movimento complexo. Por exemplo, um gesto é tocar a tela com dois dedos e movê-los simultaneamente ao redor do perímetro de um círculo abstrato para indicar rotação.

Fases Pontos distintos de tempo no fluxo do evento (por exemplo, início e término)

Stylus Instrumento (caneta) para interagir com uma tela sensível ao toque A stylus pode fornecer mais precisão do que o dedo humano. Alguns dispositivos reconhecem somente a entrada de um tipo específico de stylus. Dispositivos que reconhecem a entrada da stylus podem não reconhecer pontos múltiplos e simultâneos de contato ou o contato dos dedos.

Pressionar e tocar Um tipo específico de gesto de entrada multitoque onde o usuário pressiona o dedo contra um dispositivo sensível ao toque e, em seguida, toca com outro dedo ou dispositivo apontador. Geralmente, esse gesto é utilizado para simular um clique do botão direito do mouse em aplicativos multitoque.

A estrutura de API de entrada de toque

A API de entrada de toque ActionScript é projetada para tratar do fato que a manipulação de entrada de toque depende do ambiente de hardware e software do tempo de execução do Flash. A API de entrada de toque endereça inicialmente três necessidades de desenvolvimento de aplicativos sensíveis ao toque: descoberta, eventos e fases. Coordene essas APIs para produzir uma experiência confiável e ágil para o usuário, mesmo se o dispositivo destino for desconhecido durante o desenvolvimento do aplicativo.

Descoberta

A API de descoberta fornece a habilidade de testar o ambiente de hardware e software no tempo de execução. Os valores preenchidos pelo tempo de execução determinam a entrada de toque disponível para o tempo de execução do Flash no contexto atual. Utilize a coleção de propriedades de descoberta e métodos para configurar o aplicativo para reagir a eventos do mouse (em vez de eventos de toque no caso de alguma entrada de toque não ter suporte do ambiente). Para obter mais informações, consulte “Descoberta de suporte ao toque” na página 602

Eventos

O ActionScript gerencia os eventos de entrada de toque com os ouvintes e manipuladores de eventos, da mesma forma como faz com outros eventos. No entanto, a manipulação de evento de entrada de toque também inclui os itens a seguir:

- Um toque pode ser interpretado de diversas maneiras pelo dispositivo ou pelo sistema operacional, tanto quanto uma sequência de toques ou de forma coletiva, como um gesto.
- Um único toque em um dispositivo sensível ao toque (dedo, stylus ou dispositivo apontador) sempre envia um evento de mouse também. É possível manipular o evento de mouse com os tipos de evento na classe MouseEvent. Opcionalmente, você pode projetar seu aplicativo para somente responder a eventos de toque. Como alternativa, você pode projetar um aplicativo que responde a ambos.
- Um aplicativo pode responder a múltiplos eventos de toques simultâneos e manipular cada um deles separadamente.

Geralmente, utilize a API de descoberta para manipular condicionalmente os eventos que seu aplicativo manipula e a forma como eles são manipulados. Após o aplicativo conhecer o ambiente de tempo de execução, ele pode chamar o manipulador adequado ou estabelecer o objeto de evento correto quando o usuário interagir com o aplicativo. Opcionalmente, o aplicativo pode indicar que a entrada específica não pode ser manipulada no ambiente atual e fornecer ao usuário uma alternativa ou informação. Para obter mais informações, consulte “Tratamento de eventos de toque” na página 603 e “Tratamento de Eventos de gestos” na página 608.

Fases

Para aplicativos de toque e multitoque, objetos de evento de toque contém propriedades para monitorar as fases da interação do usuário. Crie um ActionScript para manipular fases como, por exemplo, início, atualização e fase final da entrada do usuário para fornecer feedback. Responde às fases de evento de forma que objetos visuais sejam alterados conforme o usuário toca e move o ponto de toque na tela. De forma alternativa, utilize fases para monitorar propriedades específicas de um gesto, conforme o gesto vai evoluindo.

Para eventos de ponto de toque, monitore por quanto tempo o usuário mantém pressionado um objeto interativo específico. Um aplicativo pode monitorar múltiplas fases de pontos de toque simultâneos individualmente e manipular cada um conforme necessário.

Para um gesto, interprete as informações específicas sobre a transformação de um gesto conforme ele ocorre. Monitore as coordenadas de um ponto de contato (ou diversos pontos) conforme eles se movem na tela.

Descoberta de suporte ao toque

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

Utilize as propriedades da classe Multitouchpara definir o escopo de entrada de toque manipulado por seu aplicativo. Em seguida, teste o ambiente para assegurar-se de que existe suporte para os eventos manipulados pelo ActionScript. Especificamente, primeiro estabeleça o tipo de entrada de toque de seu aplicativo. As opções são: ponto de toque, gesto ou nenhum (interpretar todas as entradas de toque como cliques do mouse e somente utilizar manipuladores de evento de mouse). Em seguida, utilize as propriedades e métodos da classe Multitouch para assegurar que o tempo de execução está em um ambiente que possui suporte à entrada de toque que seu aplicativo requer. Teste o ambiente do tempo de execução para verificar o suporte aos tipos de entrada de toque (como, por exemplo, se é possível interpretar gestos) e prossiga de acordo.

Nota: *As propriedades da classe Multitouch são propriedades estáticas e não pertencem a instâncias de nenhuma outra classe. Utilize-as com a sintaxe Multitouch.property, como no exemplo a seguir:*

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

Defina o tipo de entrada

O tempo de execução do Flash deve conhecer o tipo de entrada de toque a ser interpretado porque o evento de toque pode ter diversos elementos ou fases. Se houver o toque de um dedo em uma tela sensível ao toque, o tempo de execução envia um evento de toque? Ou ele aguarda por um gesto? Ou o tempo de execução acompanha o toque como um evento de toque de mouse? Um aplicativo que possui suporte à entrada de toque deve estabelecer o tipo dos eventos de toque que ele pode manipular para o tempo de execução do Flash. Utilize a propriedade `Multitouch.inputMode` para estabelecer o tipo de entrada de toque para o tempo de execução. O modo de entrada pode ser uma das três opções a seguir:

Nenhum Nenhuma manipulação especial é fornecida para eventos de toque. Definir: `Multitouch.inputMode=MultitouchInputMode.NONE` e utilizar a classe MouseEvent para manipular a entrada.

Pontos de toque únicos Toda a entrada de toque é interpretada individualmente e todos os pontos de toque podem ser monitorados e manipulados. Definir: `Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT` e utilizar a classe TouchEvent para manipular a entrada.

Entrada de gestos O dispositivo ou o sistema operacional interpreta a entrada como um formulário complexo ou movimento dos dedos na tela. O dispositivo ou o sistema operacional atribui de forma coletiva o movimento a um

único evento de entrada de gesto. Definir: `Multitouch.inputMode=MultitouchInputMode.GESTURE` e utilizar as classes TransformGestureEvent, PressAndTapGestureEvent ou GestureEvent para manipular a entrada.

Consulte "Tratamento de eventos de toque" na página 603 para obter um exemplo que utiliza a propriedade `Multitouch.inputMode` para definir um tipo de entrada antes de manusear um evento de toque.

Testar suporte de entrada de toque

Outras propriedades da classe Multitouch fornecem valores para ajustar seu aplicativo ao ambiente atual de suporte a toque. O tempo de execução do Flash preenche os valores do número de pontos de toque simultâneos permitidos ou gestos disponíveis. Se o tempo de execução estiver em um ambiente sem suporte à manipulação de eventos de toque, o aplicativo precisa fornecer ao usuário uma alternativa. Por exemplo, forneça a manipulação de evento de mouse ou informações sobre quais recursos estão disponíveis e indisponíveis no ambiente atual.

Você também pode utilizar a API para o suporte ao teclado, toque e mouse. Consulte "Descobrindo tipos de entrada" na página 575.

Para obter mais informações sobre o teste de compatibilidade, consulte "Solução de problemas" na página 611.

Tratamento de eventos de toque

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

Eventos básicos de toque são manipulados da mesma forma que os outros eventos como, por exemplo, eventos de mouse no ActionScript. Você pode monitorar uma série de eventos de toque definidos pelas constantes de tipo de evento na classe TouchEvent.

Nota: *Para entrada de ponto de diversos toques (tocar o dispositivo com mais de um dedo), o primeiro ponto de contato envia um evento de mouse e de toque.*

Para manipular um evento de toque básico:

1. Configure seu aplicativo para manipular eventos de toque, definindo a propriedade `flash.ui.Multitouch.inputMode` para `MultitouchInputMode.TOUCH_POINT`.
2. Anexe um ouvinte de evento à uma instância de uma classe que herda propriedades da classe InteractiveObject como, por exemplo, Sprite ou TextField.
3. Especifique o tipo de evento de toque a ser manipulado.
4. Chame a função do manipulador do evento para responder ao evento.

O exemplo a seguir exibe uma mensagem quando o quadrado desenhado em mySprite é ativado em uma tela sensível ao toque:

```
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);
}
```

Propriedade de Evento de toque

Um objeto evento é criado quando ocorre um evento. O objeto TouchEvent contém informações sobre o local e as condições do evento de toque. Você pode utilizar as propriedades do objeto de evento para obter essas informações.

Por exemplo, o código a seguir cria um objeto TouchEvent *evt* e, em seguida, exibe a propriedade `stageX` do objeto de evento (a coordenada x do ponto no espaço do Palco onde o toque ocorreu) no campo de texto:

```
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);
}
```

Consulte a classe TouchEvent para obter as propriedades disponíveis por meio do objeto de evento.

Nota: *Nem todas as propriedades TouchEvent possuem suporte em todos os ambientes de tempo de execução. Por exemplo, nem todos os dispositivos sensíveis ao toque são capazes de detectar a pressão que o usuário está aplicando na tela. Dessa forma, a propriedade `TouchEvent.pressure` não possui suporte em todos estes dispositivos. Tente testar o suporte de uma propriedade específica para verificar se o aplicativo funciona e, em seguida, consulte "Solução de problemas" na página 611 para obter mais informações.*

Fases de evento de toque

Monitore eventos de toque por meio de diversos estágios externos e sobre InteractiveObject, de forma similar a eventos de mouse. Monitore eventos de toque pelo início, meio e fim da interação de toque. A classe TouchEvent fornece valores para manipular eventos`touchBegin`, `touchMove` e `touchEnd`.

Por exemplo, é possível utilizar os eventos `touchBegin`, `touchMove` e `touchEnd` para fornecer ao usuário um feedback visual, conforme ele toca e move um objeto de exibição:

```
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;
}
```

Nota: *O ouvinte de toque inicial está anexado à mySprite, mas os ouvintes de movimento e finalização do evento de toque não estão anexados. Se o dedo do usuário ou dispositivo apontador mover-se acima do objeto de exibição, o Palco continua a monitorar o evento de toque.*

ID do ponto de toque

A propriedade `TouchEvent.touchPointID` é parte essencial da programação de aplicativos que respondem à entrada de toque. O tempo de execução do Flash atribui a cada ponto de toque, um valor `touchPointID` exclusivo. Sempre que um aplicativo responder às fases ou movimentos de entrada de toque, consulte `touchPointID` antes de manipular o evento. Os métodos de arrastas a entrada de toque da classe Sprite utilizam a propriedade `touchPointID` como parâmetro, de forma que a instância de entrada correta seja manipulada. A propriedade `touchPointID` garante que o manipulador de evento responda ao ponto de toque correto. De outra forma, o manipulador de evento responde a qualquer instância do tipo de evento de toque (como, por exemplo, todos os eventos `touchMove`) no dispositivo, criando um comportamento imprevisível. A propriedade é especialmente importante quando o usuário está arrastando objetos.

Utilize a propriedade `touchPointID` para gerenciar uma sequência inteira de toque. Uma sequência de toque possui um evento `touchBegin`, zero ou mais eventos `touchMove` e um evento `touchEnd` que possuem o mesmo valor `touchPointID`.

O exemplo a seguir estabelece a variável *touchMoveID* para testar o valor correto `touchPointID` antes de responder a um evento de movimento de toque. De outra forma, outras entradas de toque disparam o manipulador de eventos também. Observe que os ouvintes das fases de movimento e fim estão no palco mas o objeto de exibição não. O palco monitora as fases de movimento ou fim, no caso do toque do usuário mover-se além dos limites do objeto de exibição.

```
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;
}
```

Tocar e arrastar

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

Dois métodos foram adicionados à classe Sprite para fornecer suporte adicional a aplicativos sensíveis ao toque com suporte a entrada de ponto de toque: `Sprite.startTouchDrag()` e `Sprite.stopTouchDrag()`. Esses métodos comportam-se da mesma forma que `Sprite.startDrag()` e `Sprite.stopDrag()` se comportam para eventos de mouse. No entanto, observe que os métodos `Sprite.startTouchDrag()` e `Sprite.stopTouchDrag()` utilizam valores de `touchPointID` como parâmetro.

O tempo de execução atribuir o valor `touchPointID` ao objeto de evento para um evento de toque. Utilize este valor para responder a um ponto de toque específico no caso do ambiente fornecer suporte a múltiplos pontos de toque simultâneos (mesmo que não manipule gestos). Para obter mais informações sobre a propriedade `touchPointID`, consulte "ID do ponto de toque" na página 605.

O código a seguir demonstra um manipulador de evento simples de início de arrastar e fim de arrastar de um evento de toque. A variável *bg* é um objeto de exibição que contém *mySprite*:

```
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");
}
```

O exemplo a seguir mostra uma forma mais avançada de combinação de fases de arrastar com evento de toque:

```
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;
}
```

Tratamento de Eventos de gestos

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

Manipule eventos de gesto da mesma forma que eventos de toque básicos. Você pode monitorar uma série de eventos de gesto definidos pelas constantes de tipo de evento na classe TransformGestureEvent, GestureEvent e na classe PressAndTapGestureEvent.

Para manipular um evento de toque de gesto:

1. Configure o aplicativo para manipular entrada de gestos definindo a propriedade `flash.ui.Multitouch.inputMode` para `MultitouchInputMode.GESTURE`.
2. Anexe um ouvinte de evento à uma instância de uma classe que herda propriedades da classe InteractiveObject como, por exemplo, Sprite ou TextField.
3. Especifique o tipo de evento de gesto a ser manipulado.
4. Chame a função do manipulador do evento para responder ao evento.

O exemplo a seguir exibe uma mensagem quando o quadrado desenhado em *mySprite* é tocado em uma tela sensível ao toque:

```
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);
}
```

Eventos de toque de dois dedos são manipulados da mesma forma, mas utilizam a classe GestureEvent:

```
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);
}
```

Eventos de pressionar e tocar também são manipulados da mesma forma, mas utilizam a classe PressAndTapGestureEvent:

```
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);
}
```

Nota: *Nem todos os tipos de evento GestureEvent, TransformGestureEvent e PressAndTapGestureEvent possuem suporte em todos os ambientes de tempo de execução. Por exemplo, nem todos os dispositivos sensíveis ao toque são capazes de detectar um toque com diversos dedos. Dessa forma, os eventos InteractiveObject* `gestureSwipe` *não possuem suporte nestes dispositivos. Tente testar o suporte de um evento específico para verificar se o aplicativo funciona e, em seguida, consulte* *"Solução de problemas" na página 611* *para obter mais informações.*

Propriedades de Evento de gestos

Eventos de gesto possuem um escopo menor de propriedades de eventos do que eventos de toque básicos. Você pode acessá-los da mesma forma, por meio do objeto de vento na função do manipulador de eventos.

Por exemplo, o código a seguir gira a*mySprite* conforme o usuário executa um gesto de rotação nela. O campo de texto mostra o valor da rotação desde o último gesto (ao testar o código, gire-o diversas vezes para verificar a mudança do valor):

```
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();
}
```

Nota: *Nem todas as propriedades TransformGestureEvent possuem suporte em todos os ambientes de tempo de execução. Por exemplo, nem todos os dispositivos sensíveis ao toque são capazes de detectar a rotação de um gesto na tela. Dessa forma, a propriedade* `TransformGestureEvent.rotation` *não possui suporte em todos estes dispositivos. Tente testar o suporte de uma propriedade específica para verificar se o aplicativo funciona e, em seguida, consulte* *"Solução de problemas" na página 611* *para obter mais informações.*

Fases de gestos

Além disso, eventos de gesto podem ser monitorados por meio de fases, de forma que você possa acompanhar as propriedades conforme o gesto é executado. Por exemplo, é possível acompanhar as coordenadas x conforme um objeto é movido com um gesto deslizante. Utilize esses valores para desenhar uma linha por todos os pontos no caminho após o gesto deslizante ser concluído. De forma alternativa, altere visualmente um objeto de exibição conforme ele é arrastado pela tela utilizando um gesto de deslocamento. Altere o objeto novamente para concluir o gesto de deslocamento.

```
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;
}
}
```

Nota: *A frequência da fase de atualização depende do ambiente do tempo de execução. Algumas combinações de hardware e sistema operacional não podem encaminhar nenhuma atualização.*

A fase de gesto é "tudo" para eventos de gestos simples

Alguns objetos de evento de gestos não acompanham fases individuais do evento de gesto e, em vez disso, preenchem a propriedade de fase do objeto de evento com o valor "all". O gesto simples de giro e toque com dois dedos não acompanha o evento por fases múltiplas. A propriedade`phase` de um objeto de evento InteractiveObject ouvindo um evento `gestureSwipe` ou `gestureTwoFingerTap` é sempre `all`, após o evento ser despachado:

```
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"
}
```

Solução de problemas

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

O suporte de hardware e software para entrada de toque está mudando rapidamente. Esta referência não mantém uma lista de cada dispositivo que uma combinação de sistema operacional e software pode oferecer suporte a multitoque. No entanto, ela fornece orientações para utilizar a API de descoberta para determinar se o aplicativo está implementado em um dispositivo que suporte multitoque e fornece dicas para resolução de problemas no código ActionScript.

Os tempos de execução do Flash respondem a eventos de toque com base nas informações que o dispositivo, sistema operacional ou software (como, por exemplo, o navegador) passam para o tempo de execução. Esta dependência do ambiente de software complica a documentação da compatibilidade multitoque. Alguns dispositivos interpretam um gesto ou um toque de forma diferente de outro dispositivo. A rotação é definida por dois dedos girando ao mesmo tempo? A rotação de um dedo é o mesmo que desenhar um círculo na tela? Dependendo do hardware e do ambiente de software, o gesto de rotação pode ser ambos, ou algo completamente diferente. Dessa forma, o dispositivo comunica ao sistema operacional a entrada do usuário e, em seguida, o sistema operacional transmite essas informações para o tempo de execução. Se o tempo de execução estiver contido em um navegador, o software do navegador interpreta o gesto ou evento de toque e não transmite a entrada para o tempo de execução. Este comportamento é similar ao comportamento das "teclas de atalho": você tenta utilizar uma combinação específica de teclas para que o Flash execute algo dentro do navegador e, em vez disso, o navegador abre um menu.

API e classes individuais indicam se elas não compatíveis com sistemas operacionais específicos. É possível explorar entradas de API individuais aqui, começando pela classe Multitoque:
http://help.adobe.com/pt_BR/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html.

A seguir, algumas descrições de gestos e toque comuns:

Deslocar Mover um dedo da esquerda para a direita ou da direita para a esquerda. Alguns dispositivos requerem dois dedos para deslocar.

Girar Tocar com dois dedos e, em seguida, movê-los ao redor em um círculo (como se ambos estivessem traçando um círculo imaginário em uma superfície). O ponto pivô é definido como ponto intermediário entre os pontos de toque dos dois dedos.

Deslizar Mover três dedos da esquerda para a direita ou direita para a esquerda, de cima para baixo ou de baixo para cima rapidamente.

Ampliar Tocar com dois dedos e distanciá-los para ampliar e aproximá-los para reduzir.

Pressionar e tocar Mover ou pressionar um dedo e, em seguida, tocar a superfície com outro dedo.

Cada dispositivo possui sua própria documentação sobre os gestos que o dispositivo suporta e como executar cada gesto no dispositivo. Geralmente, o usuário deve remover todos os dedos do dispositivo entre gestos, dependendo do sistema operacional.

Se você perceber que seu aplicativo não está respondendo a eventos de toque ou gestos, execute os testes a seguir:

1. Existem ouvintes de evento para eventos de gesto ou toque anexados à uma classe de objeto que herda atributos da classe InteractiveObject? Somente instâncias InteractiveObjects pode monitorar eventos de gestos ou toques
2. O aplicativo está sendo testado com o Flash Professional CS5? Em caso afirmativo, tente publicar e testar o aplicativo, porque o Flash Professional pode interceptar as interações.
3. Começe de forma simplificada e verifique o que funciona (o código de exemplo a seguir é da entrada de API para `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);
}
```

Toque no retângulo. Se este exemplo funcionar, você saberá que o ambiente tem suporte a toque simples, Dessa forma, você poderá testar manipulações maiis complexas.

O teste para suporte de gestos é mais complicado. Um dispositivo individual ou sistema operacional possuem suporte a uma combinação de entrada de gestos ou nenhuma combinação.

A seguir, um teste simples para o gesto de zoom:

```
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";
}
```

Execute um teste de gesto no dispositivo e verifique se o campo de texto é populado com a mensagem `Zoom is supported`. O ouvinte de evento é adicionado ao palco, de forma que você possa executar o gesto em qualquer parte do aplicativo.

A seguir, um teste simples para o gesto de panorâmico:

```
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";
}
```

Execute um teste de gesto panorâmico no dispositivo e verifique se o campo de texto é populado com a mensagem `Pan is supported`. O ouvinte de evento é adicionado ao palco, de forma que você possa executar o gesto em qualquer parte do aplicativo.

Alguns sistemas operacionais e combinações de dispositivos possuem suporte a ambos gêneros, outros possuem suporte a somente um gênero e outros a nenhum. Teste o ambiente de desenvolvimento de seu aplicativo para certificar-se.

Problemas conhecidos

Os itens a seguir são problemas conhecidos relacionados à entrada de toque:

1. Mobile Internet Explorer no sistema operacional Windows Mobile faz o zoom automático de conteúdo de arquivo SWF:

   Este comportamento do zoom do Internet Explorer pode ser sobrescrito, adicionando o código a seguir à página HTML do arquivo SWF:

   ```
   <head>
   <meta name="viewport" content="width=device-width, height=device-height, initial-
   scale=1.0">
   </head>
   ```

2. No Windows 7 (e possivelmente, em outros sistemas operacionais), o usuário deve levantar o dispositivo apontador (ou os dedos) da tela entre gestos. Por exemplo, para girar e ampliar uma imagem:
   - Execute o gesto de girar.
   - Levante o dedo da tela.
   - Posicione os dedos de volta à tela e execute o gesto de ampliação.
3. No Windows 7 (e possivelmente, outros sistemas operacionais), os gestos de girar e ampliar nem sempre geram uma fase de atualização se o usuário executar o gesto de forma muito rápida.
4. O Windows 7 Starter Edition não possui suporte a multitoque. Consulte o fórum do AIR Labs para obter mais detalhes: http://forums.adobe.com/thread/579180?tstart=0
5. Para o Mac OS 10.5.3 e superior, o valores `Multitouch.supportsGestureEvents` é sempre `true`, mesmo que o hardware não possua suporte a eventos de gestos.

Capítulo 33: Copiar e colar

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Use as classes na API da área de transferência para copiar informações para e da área de transferência do sistema. Os formatos de dados que podem ser transferidos de/para um aplicativo executado no Adobe® Flash® Player ou no Adobe® AIR® são os seguintes:

- Texto
- Texto formatado em HTML
- Dados em RTF
- Objetos serializados
- Referências de objetos (válido apenas dentro do aplicativo originador)
- Bitmaps (somente AIR)
- Arquivos (somente AIR)
- Strings de URLs (somente AIR)

Princípios básicos do copiar e colar

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

A API de copiar e colar contém as seguintes classes.

Pacote	Classes
flash.desktop	• Clipboard • ClipboardFormats • ClipboardTransferMode

A propriedade estática `Clipboard.generalClipboard` representa a área de transferência do sistema operacional. A classe Clipboard fornece métodos para ler e escrever dados em objetos da área de transferência.

As classes HTMLLoader (no AIR) e TextField implementam um comportamento padrão para os atalhos comuns do teclado para copiar e colar. Para implementar o comportamento do atalho de copiar e colar para componentes personalizados, você pode ouvir esses pressionamentos de tecla diretamente. Você também pode usar comandos de menu nativos juntamente com equivalentes de teclas para responder aos pressionamentos de tecla indiretamente.

Diferentes representações das mesmas informações podem ser disponibilizadas em um único objeto Clipboard para aumentar a capacidade de outros aplicativos entenderem e usarem os dados. Por exemplo, uma imagem pode ser incluída como dados de imagem, como um objeto Bitmap serializado e como um arquivo. O processamento dos dados em um formato pode ser adiado de forma que o formato não seja realmente criado até que os dados naquele formato sejam lidos.

Leitura e gravação na área de transferência do sistema

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Para ler a área de transferência do sistema operacional, chame o método `getData()` do objeto `Clipboard.generalClipbooard`, transmitindo o nome do formato a ser lido:

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
    var text:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);
}
```

Nota: *O conteúdo em execução no Flash Player ou em uma caixa de proteção "não aplicativo" no AIR pode chamar apenas o método `getData()` em um manipulador de eventos para um evento `paste`. Em outras palavras, somente os códigos em execução na caixa de proteção do aplicativo do AIR podem chamar o método `getData()` fora de um manipulador de eventos `paste`.*

Para escrever na área de transferência, adicione os dados ao objeto `Clipboard.generalClipboard` em um ou mais formatos. Qualquer dado existente no mesmo formato é substituído automaticamente. No entanto, é uma boa prática limpar também a área de transferência do sistema antes de escrever novos dados nela para certificar-se de que os dados não relacionados em nenhum outro formato também sejam excluídos.

```
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);
```

Nota: *O conteúdo em execução no Flash Player ou em uma caixa de proteção "não aplicação" no AIR pode chamar apenas o método `setData()` em um manipulador de eventos para um evento de usuário, como eventos de teclado ou do mouse, além de um evento `copy` ou `cut`. Em outras palavras, somente os códigos em execução na caixa de proteção do aplicativo do AIR podem chamar o método `setData()` fora de um manipulador de eventos do usuário.*

Copiar e colar HTML no AIR

Adobe AIR 1.0 e posterior

O ambiente HTML do Adobe AIR fornece seu próprio conjunto de eventos e comportamento padrão para copiar e colar. Apenas códigos em execução na caixa de proteção do aplicativo podem acessar a área de transferência do sistema diretamente pelo objeto do AIR `Clipboard.generalClipboard`. Códigos JavaScript em uma caixa de proteção de não aplicativo podem acessar a área de transferência pelo objeto de evento despachado em resposta a um dos eventos de copiar ou colar despachados por um elemento em um documento HTML.

Os eventos de copiar e colar incluem: `copy`, `cut` e `paste`. O objeto despachado para esses eventos fornece acesso à área de transferência pela propriedade `clipboardData`.

Comportamento padrão

Adobe AIR 1.0 e posterior

Por padrão, o AIR copia itens selecionados em resposta ao comando de copiar, que pode ser gerado por um atalho do teclado ou um menu de contexto. Em regiões editáveis, o AIR recorta textos em resposta ao comando de recortar ou cola textos para o cursor ou seleção em resposta ao comando de colar.

Para impedir o comportamento padrão, seu manipulador de eventos pode chamar o método `preventDefault()` do objeto de evento despachado.

Uso da propriedade clipboardData do objeto de evento

Adobe AIR 1.0 e posterior

A propriedade `clipboardData` do objeto de evento despachado como um resultado de um dos eventos de copiar ou colar permite que você leia e escreva dados da área de transferência.

Para escrever na área de transferência ao manipular um evento de copiar ou recortar, use o método `setData()` do objeto `clipboardData`, transmitindo os dados para copiar e o tipo MIME:

```
function customCopy(event){
    event.clipboardData.setData("text/plain", "A copied string.");
}
```

Para acessar os dados que estão sendo colados, você pode usar o método `getData()` do objeto `clipboardData`, transmitindo o tipo MIME do formato de dados. Os formatos disponíveis são relatados pela propriedade `types`.

```
function customPaste(event){
    var pastedData = event.clipboardData("text/plain");
}
```

O método `getData()` e a propriedade `types` podem apenas ser acessados no objeto de evento despachado pelo evento `paste`.

O exemplo a seguir ilustra como substituir o comportamento de copiar e colar padrão em uma página HTML. O manipulador de eventos `copy` coloca em itálico o texto copiado e o copia para a área de transferência como texto HTML. O manipulador de eventos `cut` copia os dados selecionados para a área de transferência e os remove do documento. O manipulador `paste` insere o conteúdo da área de transferência como HTML e aplica um estilo na inserção como texto em negrito.

```
<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>
```

Formatos de dados da área de transferência

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Os formatos da área de transferência descrevem os dados colocados em um objeto Clipboard. O Flash Player ou o AIR traduz automaticamente os formatos de dados padrão entre tipos de dados do ActionScript e formatos da área de transferência do sistema. Além disso, os objetos do aplicativo podem ser transferidos internamente em ou entre aplicativos com base no ActionScript, usando formatos definidos pelo aplicativo.

Um objeto Clipboard pode conter representações das mesmas informações em diferentes formatos. Por exemplo, um objeto Clipboard representando um Sprite poderia incluir um formato de referência para ser usado no mesmo aplicativo, um formato serializado para ser usado por outro aplicativo em execução no Flash Player ou no AIR, um formato bitmap para ser usado por um editor de imagens e um formato de lista de arquivo, talvez com renderização adiada para codificar um arquivo PNG, para copiar ou arrastar uma representação do Sprite para o sistema de arquivos.

Formatos de dados padrão

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

As constantes que definem os nomes de formato padrão são fornecidas na classe ClipboardFormats:

Constante	Descrição
TEXT_FORMAT	Dados em formato de texto são traduzidos para e da classe String do ActionScript.
HTML_FORMAT	Texto com marcação HTML.
RICH_TEXT_FORMAT	Dados em RTF são traduzidos para e da classe ByteArray do ActionScript. A marcação RTF não é interpretada ou traduzida de forma alguma.
BITMAP_FORMAT	(somente AIR) Dados em formato bitmap são traduzidos para e da classe BitmapData do ActionScript.
FILE_LIST_FORMAT	(somente AIR) Dados em formato de lista de arquivo são traduzidos para e de uma matriz de objetos File do ActionScript.
URL_FORMAT	(somente AIR) Dados em formato de URL são traduzidos para e a partir da classe String do ActionScript.

Ao copiar e colar dados em resposta a um evento `copy`, `cut` ou `paste` em conteúdo HTML hospedado em um aplicativo do AIR, devem ser usados tipos MIME em vez das sequências de caracteres ClipboardFormat. Os tipos MIME de dados válidos são:

Tipo MIME	Descrição
Texto	"text/plain"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
Lista de arquivos	"application/x-vnd.adobe.air.file-list"

Nota: *Dados RTF não estão disponíveis da propriedade `clipboardData` do objeto de evento despachado como resultado de um evento `paste` no conteúdo HTML.*

Formatos de dados personalizados

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Você pode usar formatos personalizados definidos por aplicativo para transferir objetos como referências ou cópias serializadas. As referências são válidas apenas no mesmo aplicativo. Os objetos serializados podem ser transferidos entre aplicativos, mas só podem ser usados com objetos que permanecem válidos quando serializados e desserializados. Geralmente, objetos podem ser serializados se suas propriedades forem tipos simples ou objetos serializável.

Para adicionar um objeto serializado a um objeto Clipboard, defina o parâmetro serializable como *serializable* como `true` ao chamar o método `Clipboard.setData()`. O nome do formato pode ser um dos formatos padrão ou uma sequência de caracteres arbitrária definida pelo seu aplicativo.

Modos de transferência

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Quando um objeto é escrito na área de transferência usando um formato de dados personalizado, os dados do objeto podem ser lidos da área de transferência como referência ou uma cópia serializada do objeto original. Há quatro modos de transferência que determinam se os objetos são transferidos como referências ou como cópias serializadas:

Modo de transferência	Descrição
ClipboardTransferModes.ORIGINAL_ONLY	Apenas uma referência é retornada. Se nenhuma referência estiver disponível, um valor null será retornado.
ClipboardTransferModes.ORIGINAL_PREFFERED	Uma referência é retornada, se disponível. Caso contrário, uma cópia serializada é retornada.
ClipboardTransferModes.CLONE_ONLY	Apenas uma cópia serializada é retornada. Se nenhuma cópia serializada estiver disponível, será retornado um valor null.
ClipboardTransferModes.CLONE_PREFFERED	Uma cópia serializada é retornada, se disponível. Caso contrário, uma referência é retornada.

Leitura e escrita de formatos de dados personalizados

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Ao gravar um objeto na área de transferência, você pode usar qualquer sequência de caracteres que não comece com os prefixos reservados `air:` ou `flash:` para o parâmetro *format*. Use a mesma sequência de caracteres do formato para ler o objeto. Os exemplos a seguir ilustram como ler e escrever objetos na área de transferência:

```
public function createClipboardObject(object:Object):Clipboard{
    var transfer:Clipboard = Clipboard.generalClipboard;
    transfer.setData("object", object, true);
}
```

Para extrair um objeto serializado do objeto da área de transferência (após uma operação de soltar ou colar), use o mesmo nome de formato e os modos de transferência `CLONE_ONLY` ou `CLONE_PREFFERED`.

```
var transfer:Object = clipboard.getData("object", ClipboardTransferMode.CLONE_ONLY);
```

Uma referência é sempre adicionada ao objeto Clipboard. Para extrair a referência do objeto da área de transferência (após uma operação de soltar ou colar), em vez da cópia serializada, use os modos de transferência `ORIGINAL_ONLY` ou `ORIGINAL_PREFFERED`:

```
var transferredObject:Object =
    clipboard.getData("object", ClipboardTransferMode.ORIGINAL_ONLY);
```

As referências são válidas apenas se o objeto Clipboard se originar do aplicativo atual. Use o modo de transferência `ORIGINAL_PREFFERED` para acessar a referência quando ela estiver disponível e o clone serializado quando a referência não estiver disponível.

Renderização adiada

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

Se criar um formato de dados for computacionalmente caro, você poderá usar o processamento adiado fornecendo uma função que forneça os dados em demanda. A função só é chamada se um receptor da operação de soltar ou colar solicitar dados no formato adiado.

A função de processamento é adicionada a um objeto Clipboard usando o método `setDataHandler()`. A função deve retornar os dados no formato apropriado. Por exemplo, se você chamou `setDataHandler(ClipboardFormat.TEXT_FORMAT, writeText)`, a função `writeText()` deverá retornar uma sequência de caracteres.

Se um formato de dados do mesmo tipo for adicionado a um objeto Clipboard com o método `setData()`, esses dados tem precedência sobre a versão adiada (a função de renderização nunca é chamada). A função de renderização pode ou não ser chamada novamente se os mesmos dados da área de transferência forem acessados uma segunda vez.

Nota: *No Mac OS X, a renderização adiada só funciona com formatos de dados personalizados. No caso de formatos de dados padrão, a função de renderização é chamada imediatamente.*

Colagem de texto usando uma função de renderização adiada

Flash Player 10 e posterior, Adobe AIR 1.0 e posterior

O exemplo a seguir ilustra como implementar uma função de renderização adiada.

Quando um usuário pressiona o botão Copiar, o aplicativo limpa a área de transferência para garantir que nenhum dado de operações anteriores seja mantido. Em seguida, o método `setDataHandler()` define a função `renderData()` como renderizador da área de transferência.

Quando um usuário seleciona o comando Colar no menu de contexto do campo de texto de destino, o aplicativo acessa a área de transferência e define o texto de destino. Como o formato de dados de texto na área de transferência foi definido com uma função em vez de uma string, a área de transferência chama a função `renderData()`. A função `renderData()` retorna o texto no texto de origem, que é então atribuído ao texto de destino.

Observe que se você editar o texto de origem antes de pressionar o botão Colar, a edição será refletida no texto colado, mesmo quando a edição ocorrer após o botão de copiar ter sido pressionado. Isso porque a função de processamento não copia o texto de origem até que o botão de colar seja pressionado. (Ao usar o processamento adiado em um aplicativo real, você pode desejar armazenar ou proteger os dados de origem de alguma maneira para impedir esse problema.)

Exemplo de Flash

```
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;
            }
        }
    }
}
```

Exemplo do Flex

```
<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>
```

Capítulo 34: Entrada do acelerômetro

Flash Player 10.1 e posterior, Adobe AIR 2 e posterior

A classe Accelerometer despacha eventos com base na atividade detectada pelo sensor de movimento do dispositivo. Estes dados representam o local do dispositivo ou movimento ao longo de um eixo de tridimensional. Quando o dispositivo se move, o sensor detecta este movimento e retorna as coordenadas de aceleração do dispositivo. A classe Accelerometer fornece métodos para consultar se o acelerômetro é suportado ou não e também para definir a taxa na qual os eventos de aceleração são despachados.

Os eixos do acelerômetro são normalizados na orientação do visor, e não na orientação física do dispositivo. Quando o dispositivo reorienta o visor, os eixos do acelerômetro também são reorientados. Assim, o eixo Y está sempre aproximadamente na posição vertical quando o usuário segura o telefone na posição vertical normal de exibição, independentemente da maneira como o telefone é girado. Se a orientação automática estiver desligada, por exemplo quando o conteúdo SWF em um navegador está no modo de tela cheia, os eixos do acelerômetro não serão reorientados quando o dispositivo for girado.

Mais tópicos da Ajuda

flash.sensors.Accelerometer

flash.events.AccelerometerEvent

Verificando suporte ao acelerômetro

Utilize a propriedade `Accelerometer.isSupported` para testar o ambiente de tempo de execução em relação a habilidade de utilizar este recurso:

```
if (Accelerometer.isSupported)
{
    // Set up Accelerometer event listeners and code.
}
```

A classe acelerômetro e seus membros são acessíveis pelas versões de tempo de execução listados em cada entrada de API. No entanto o ambiente atual durante o tempo de execução determina a disponibilidade para este recurso. Por exemplo, você pode compilar o código as propriedades da classe Accelerometer para o Flash Player 10.1, mas será necessário utilizar a propriedade `Accelerometer.isSupported` para testar a disponibilidade do recurso Accelerometer no dispositivo do usuário. Caso `Accelerometer.isSupported` seja `true` durante o tempo de execução, então o suporte ao Accelerometer existe.

Detectando mudanças no acelerômetro

Para utilizar o sensor de acelerômetro, instancie um objeto Accelerometer e registre para `atualizar` eventos enviados. O evento `update` é um objeto de evento Accelerometer. O evento possui quatro propriedades, são eles:

- `accelerationX`—Aceleração ao longo do eixo x, medida em g. O eixo x vai da esquerda para a direita do dispositivo quando está na posição perpendicular. (O dispositivo está na vertical quando o topo do dispositivo está voltado para cima). A aceleração será positiva se o dispositivo for movido para a direita.

- `accelerationY`—Aceleração ao longo do eixo y, medida em g. O eixo y percorre a parte superior do dispositivo quando está na posição perpendicular. (O dispositivo está na vertical quando o topo do dispositivo está voltado para cima). A aceleração será positiva se o dispositivo for movido para cima deste eixo.
- `accelerationZ`—Aceleração ao longo do eixo z, medida em g. O eixo z corre perpendicularmente à parte fronta do dispositivo. A aceleração é positiva se você mover o dispositivo de forma que a face do dispositivo aponte para cima. A aceleração é negativa se a face do dispositivo apontar para o chão.
- `timestamp`—O número de milissegundos no momento do evento a partir do início da execução.

1 g é a aceleração padrão da gravidade, aproximadamente 9.8 m/s^2.

Este é um exemplo básico que exibe dados de um acelerômetro em um campo de texto:

```
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()
}
```

Para utilizar este exemplo, certifique-se de criar o campo de texto `accTextField` e adicioná-lo para a lista de exibição antes de utilizar este código.

Você pode ajustar o intervalo de tempo para eventos de acelerômetro chamando o método `setRequestedUpdateInterval()` do objeto Accelerometer. Este método usa um parâmetro `interval`, que é o intervalo de atualização solicitado em milisegundos:

```
var accl:Accelerometer;
accl = new Accelerometer();
accl.setRequestedUpdateInterval(1000);
```

A hora real entre atualizações do acelerômetro pode ser maior ou menor do que este valor. Qualquer mudança no intervalo de atualização afeta todos os ouvintes registrados. Se o método `setRequestedUpdateInterval()` não for chamado, o aplicativo recebe atualizações com base no intervalo padrão do dispositivo.

Dados do acelerômetro tem alguns graus de imprecisão. Você pode utilizar uma média de movimento de dados recentes para aperfeiçoar os dados. Por exemplo, o exemplo a seguir fatora leituras recentes do acelerômetro com a leitura atual para obter um resultado arredondado:

```
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));
}
```

No entanto, esta média de movimentação só é desejável se o intervalo de atualização do acelerômetro for pequeno.

Capítulo 35: Arrastar e soltar no AIR

Adobe AIR 1.0 e posterior

Use as classes na API de arrastar e soltar do Adobe® AIR™ para suportar gestos de arrastar e soltar de interface do usuário. Um *gesto* neste sentido é uma ação do usuário, mediada pelo sistema operacional e seu aplicativo, expressando uma intenção de copiar, mover ou vincular informações. Um gesto de *arrastar para fora* ocorre quando o usuário arrasta um objeto para fora de um componente ou aplicativo. Um gesto de *arrastar para dentro* ocorre quando o usuário arrasta para dentro um objeto de fora um componente ou aplicativo.

Com a API de arrastar e soltar, você pode permitir que um usuário arraste dados entre aplicativos e entre componentes dentro de um aplicativo. Os formatos de transferência suportados incluem:

- Bitmaps
- Arquivos
- Texto formatado em HTML
- Texto
- Dados em RTF
- URLs
- Promessas de arquivo
- Objetos serializados
- Referências a objetos (válido somente dentro do aplicativo originador)

Princípios básicos de arrastar e soltar no AIR

Adobe AIR 1.0 e posterior

Para ver uma rápida explicação e exemplos de código do uso do arrastar e soltar em um aplicativo do AIR, consulte os seguintes artigos de início rápido no Adobe Developer Connection:

- Dando suporte ao arrastar e soltar e ao copiar e colar (Flex)
- Dando suporte ao arrastar e soltar e ao copiar e colar (Flash)

A API de arrastar e soltar contém as seguintes classes:

Pacote	Classes
flash.desktop	• NativeDragManager • NativeDragOptions • Clipboard • URLFilePromise • IFilePromise Constantes usadas com a API de arrastar e soltar são definidas nas seguintes classes: • NativeDragActions • ClipboardFormat • ClipboardTransferModes
flash.events	NativeDragEvent

Estágios do gesto de arrastar e soltar

O gesto de arrastar e soltar possui três estágios:

Iniciação *Um usuário inicia uma operação de arrastar e soltar arrastando de um componente, ou um item em um componente, enquanto mantém pressionado o botão do mouse.* O componente que é a origem do item arrastado é normalmente designado como o iniciador do ação de arrastar e despacha eventos `nativeDragStart` e `nativeDragComplete`. Um aplicativo do Adobe AIR inicia uma operação de arrastar chamando o método `NativeDragManager.doDrag()` em resposta a um evento `mouseDown` ou `mouseMove`.

Se a operação de arrastar for iniciada de fora do aplicativo do AIR, não haverá nenhum objeto iniciador para despachar os eventos `nativeDragStart` ou `nativeDragComplete`.

Arrastar *Ao segurar o botão do mouse, o usuário move o cursor do mouse para outro componente, aplicativo ou a área de trabalho.* Uma vez que a ação de arrastar esteja em andamento, o objeto iniciador despacha eventos `nativeDragUpdate`. (No entanto, este evento não é despachado no AIR for Linux.) Quando o usuário move o mouse sobre um possível destino no qual soltar em um aplicativo do AIR, o destino despacha um evento `nativeDragEnter`. O manipulador de eventos pode inspecionar o objeto de evento para determinar se os dados arrastados estão disponíveis em um formato que o destino aceite e, se sim, permitir ao usuário soltar os dados nele chamando o método `NativeDragManager.acceptDragDrop()`.

Enquanto o gesto de arrastar permanece sobre um objeto interativo, esse objeto despacha eventos `nativeDragOver`. Quando o gesto de arrastar deixa o objeto interativo, ele despacha um evento `nativeDragExit`.

Soltar *O usuário libera o mouse sobre um destino de soltar aceitável.* Se o destino for um componente ou aplicativo do AIR, o objeto de destino despacha um evento `nativeDragDrop`. O manipulador de eventos pode acessar os dados transferidos do objeto de evento. Se o destino estiver fora do AIR, o sistema operacional ou outro aplicativo manipulará a ação de soltar. Nos dois casos, o objeto iniciador despacha um evento `nativeDragComplete` (se a ação de arrastar tiver iniciado de dentro do AIR).

A classe NativeDragManager controla os gestos de arrastar para dentro e para fora. Todos os membros da classe NativeDragManager são estáticos, não crie uma instância dessa classe.

O objeto Clipboard

Os dados arrastados para dentro ou para fora de um aplicativo ou componente são contidos em um objeto Clipboard. Um único objeto Clipboard pode disponibilizar representações diferentes das mesmas informações para aumentar a probabilidade de que outro aplicativo possa entender e usar os dados. Por exemplo, uma imagem poderia ser incluída como dados de imagem, como um objeto Bitmap serializado e como um arquivo. O processamento dos dados em um formato pode ser adiado para uma função de processamento não chamada até que os dados sejam lidos.

Após um gesto de arrastar ter sido iniciado, o objeto Clipboard pode apenas ser acessado de dentro de um manipulador de eventos para os eventos `nativeDragEnter`, `nativeDragOver` e `nativeDragDrop`. Após o gesto de arrastar ter sido finalizado, o objeto Clipboard não poderá ser lido ou reutilizado.

Um objeto de aplicativo pode ser transferido como uma referência e um objeto serializado. As referências são válidas apenas dentro do aplicativo originador. Transferências de objetos serializados são válidas entre aplicativos AIR, mas podem apenas ser usadas com objetos que permanecem válidos quando serializados e desserializados. Objetos serializados são convertidos no Action Message Format para ActionScript 3 (AMF3), um formato de transferência de dados baseado em sequência de caracteres.

Trabalho com a estrutura do Flex

Na maioria dos casos, é melhor usar a API de arrastar e soltar do Adobe® Flex™ ao criar aplicativos do Flex. A estrutura do Flex fornece um conjunto de recursos equivalente quando um aplicativo do Flex é executado no AIR (ela usa o NativeDragManager do AIR internamente). O Flex também mantém um conjunto de recursos mais limitado quando um aplicativo ou componente está em execução dentro do ambiente do navegador mais restritivo. As classes do AIR não podem ser usadas em componentes ou aplicativos executados fora do ambiente de tempo de execução do AIR.

Suporte ao gesto de arrastar para fora

Adobe AIR 1.0 e posterior

Para suportar o gesto de arrastar para fora, você deve criar um objeto Clipboard em resposta a um evento `mouseDown` e enviá-lo ao método `NativeDragManager.doDrag()`. Seu aplicativo pode então ouvir o evento `nativeDragComplete` sobre o objeto iniciador para determinar que ação tomar quando o usuário conclui ou abandona o gesto.

Preparação de dados para transferência

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para preparar dados ou um objeto para arrastar, crie um objeto Clipboard e adicione as informações a serem transferidas em um ou mais formatos. Você pode usar os formatos de dados padrão para passar dados, que podem ser convertidos automaticamente em formatos nativos da área de transferência e em formatos definidos pelo aplicativo para passar objetos.

Se, do ponto de vista computacional, for caro converter as informações a serem transferidas para um formato específico, você pode fornecer o nome de uma função de manipulador para realizar a conversão. A função é chamada se, e apenas se, o componente ou aplicativo receptor ler o formato associado.

Para mais informações sobre formatos de área de transferência, veja “Formatos de dados da área de transferência” na página 617.

O exemplo a seguir ilustra como criar um objeto Clipboard contendo um bitmap em vários formatos: um objeto Bitmap, um formato de bitmap nativo e um formato da lista de arquivo contendo o arquivo do qual o bitmap foi originalmente carregado:

```
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;
}
```

Início de uma operação de arrastar e soltar

Adobe AIR 1.0 e posterior

Para iniciar uma operação de arrastar, chame o método `NativeDragManager.doDrag()` em resposta a um evento de mouse para baixo. O método `doDrag()` é um método estático que adota os seguintes parâmetros:

Parâmetro	Descrição
iniciador	O objeto do qual a ação de arrastar se origina e que despacha os eventos `dragStart` e `dragComplete`. O iniciador deve ser um objeto interativo.
área de transferência	O objeto Clipboard contendo os dados a serem transferidos. O objeto Clipboard é referenciado nos objetos NativeDragEvent despachados durante a sequência de arrastar e soltar.
dragImage	(Opcional) Um objeto BitmapData a ser exibido durante a ação de arrastar. A imagem pode especificar um valor `alpha`. (Observação: o Microsoft Windows sempre aplica uma atenuação alfa fixa a imagens de arrastar).
deslocamento	(Opcional) Um objeto Point especificando o deslocamento da imagem de arrastar do ponto ativo do mouse. Use coordenadas negativas para mover a imagem de arrastar para cima e à esquerda com relação ao cursor do mouse. Se nenhum deslocamento for fornecido, o canto superior esquerdo da imagem de arrastar será posicionado no ponto ativo do mouse.
actionsAllowed	(Opcional) Um objeto NativeDragOptions especificando que ações (copiar, mover ou vincular) são válidas para a operação de arrastar. Se nenhum argumento for fornecido, todas as ações serão permitidas. O objeto DragOptions é referenciado em objetos NativeDragEvent para permitir que um possível destino da ação de arrastar verifique se as ações permitidas são compatíveis com o objetivo do componente de destino. Por exemplo, um componente “trash” pode aceitar apenas gestos de arrastar que permitem a ação de mover.

O exemplo a seguir ilustra como iniciar uma operação de arrastar para um objeto de bitmap carregado de um arquivo. O exemplo carrega uma imagem e, em um evento `mouseDown`, inicia a operação de arrastar.

```
 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);
    }
}
}
```

Conclusão de uma transferência de arrastar para fora

Adobe AIR 1.0 e posterior

Quando um usuário solta o item arrastado liberando o mouse, o objeto iniciador despacha um evento `nativeDragComplete`. Você pode verificar a propriedade `dropAction` do objeto de evento e executar a ação apropriada. Por exemplo, se a ação for `NativeDragAction.MOVE`, você poderia remover o item de origem de seu local original. O usuário pode abandonar um gesto de arrastar liberando o botão do mouse enquanto o cursor está fora de um destino de arrastar aceitável. O gerente de arrastar define a propriedade `dropAction` para um gesto abandonado a `NativeDragAction.NONE`.

Suporte ao gesto de arrastar para dentro

Adobe AIR 1.0 e posterior

Para suportar o gesto de arrastar para dentro, seu aplicativo (ou, mais tipicamente, um componente visual do seu aplicativo) deve responder aos eventos `nativeDragEnter` ou `nativeDragOver`.

Etapas em uma operação típica de arrastar

Adobe AIR 1.0 e posterior

A seguinte sequência de eventos é típica para uma operação de arrastar:

1. O usuário arrasta um objeto da área de transferência sobre um componente.
2. O componente despacha um evento `nativeDragEnter`.
3. O manipulador de eventos `nativeDragEnter` examina o objeto de evento para verificar os formatos de dados disponíveis e as ações permitidas. Se o componente pode manipular a ação de soltar, ele chama `NativeDragManager.acceptDragDrop()`.

4 O NativeDragManager altera o cursor do mouse para indicar que o objeto pode ser solto.

5 O usuário solta o objeto sobre o componente.

6 O componente receptor despacha um evento `nativeDragDrop`.

7 O componente receptor lê os dados no formato desejado do objeto Clipboard dentro do objeto de evento.

8 Se o gesto de arrastar foi originado no aplicativo do AIR, o objeto interativo iniciador despacha um evento `nativeDragComplete`. Se o gesto foi originado fora do AIR, nenhum comentário é enviado.

Reconhecimento de um gesto de arrastar para dentro

Adobe AIR 1.0 e posterior

Quando um usuário arrasta um item da área de transferência para dentro dos limites de um componente visual, o componente despacha eventos `nativeDragEnter` e `nativeDragOver`. Para determinar se o componente pode aceitar o item da área de transferência, os manipuladores para esses eventos pode verificar as propriedades `clipboard` e `allowedActions` do objeto de evento. Para sinalizar que o componente pode aceitar a ação de soltar, o manipulador de eventos deve chamar o método `NativeDragManager.acceptDragDrop()`, transmitindo uma referência ao componente receptor. Se mais de um ouvinte de evento registrado chamar o método `acceptDragDrop()`, o último manipulador da lista tem precedência. A chamada `acceptDragDrop()` permanece válida até que o mouse deixe os limites do objeto recebedor, disparando o evento `nativeDragExit`.

Se mais de uma ação for permitida no parâmetro `allowedActions` transmitido a `doDrag()`, o usuário pode indicar qual das ações permitidas pretende mantendo pressionada uma tecla de modificador. O gerente de arrastar altera a imagem do cursor para dizer ao usuário que ação ocorreria se ele concluísse a ação de soltar. A ação pretendida é reportado pela propriedade `dropAction` do evento NativeDragEvent. O conjunto de ação para um gesto de arrastar é apenas consultivo. Os componentes envolvidos na transferência devem implementar o comportamento apropriado. Para completar uma ação de mover, por exemplo, o iniciador de arrastar poderia remover o item arrastado e o destino de soltar poderia adicioná-lo.

Seu destino de arrastar pode limitar a ação de soltar para uma das três ações possíveis definindo a propriedade `dropAction` da classe NativeDragManager. Se um usuário tenta escolher uma ação diferente usando o teclado, NativeDragManager exibe o cursor *unavailable*. Define a propriedade `dropAction` nos manipuladores para os eventos `nativeDragEnter` e `nativeDragOver`.

O exemplo a seguir ilustra um manipulador de eventos para um evento `nativeDragEnter` ou `nativeDragOver`. Esse manipulador aceita apenas um gesto de arrastar para dentro se a área de transferência sendo arrastada contiver dados em formato de texto.

```
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
    }
}
```

Conclusão da ação de soltar

Adobe AIR 1.0 e posterior

Quando o usuário solta um item arrastado em um objeto interativo que aceitou o gesto, o objeto interativo despacha um evento `nativeDragDrop`. O manipulador desse evento pode extrair os dados a propriedade `clipboard` do objeto de evento.

Quando a área de transferência contém um formato definido por aplicativo, o parâmetro `transferMode` transmitido ao método `getData()` do objeto Clipboard determina se o gerenciador de arrastar retorna uma referência ou uma versão serializada do objeto.

O exemplo a seguir ilustra um manipulador de eventos para o evento `nativeDragDrop`:

```
 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));
}
```

Depois que o manipulador de eventos for encerrado, o objeto Clipboard não será mais válido. Qualquer tentativa de acessar o objeto ou seus dados gera um erro.

Atualização da aparência visual de um componente

Adobe AIR 1.0 e posterior

Um componente pode atualizar sua aparência visual com base nos eventos NativeDragEvent. A tabela a seguir descreve os tipos de alterações que um componente típico faria em resposta aos diferentes eventos:

Evento	Descrição
nativeDragStart	O objeto interativo iniciante pode usar o evento `nativeDragStart` para fornecer comentários visuais que o gesto de arrastar originou desse objeto interativo.
nativeDragUpdate	O objeto interativo iniciante pode usar o evento nativeDragUpdate para atualizar seu estado durante o gesto. (Este evento não existe no AIR for Linux.)
nativeDragEnter	Um possível objeto interativo receptor pode usar esse evento para assumir o foco ou indicar visualmente que ele pode ou não aceitar a ação de soltar.
nativeDragOver	Um possível objeto interativo receptor pode usar esse evento para responder ao movimento do mouse dentro do objeto interativo, como quando o mouse insere uma região “ativa” de um componente complexo, como uma exibição de mapa de rua.
nativeDragExit	Um possível objeto interativo receptor pode usar esse evento para restaurar seu estado quando um gesto de arrastar se move para fora de seus limites.
nativeDragComplete	O objeto interativo iniciante pode usar esse evento para atualizar seu modelo de dados associado, como removendo um item de uma lista e para restaurar seu estado visual.

Rastreamento da posição do mouse durante um gesto de arrastar para dentro

Adobe AIR 1.0 e posterior

Enquanto um gesto de arrastar permanece sobre um componente, esse componente despacha eventos `nativeDragOver`. Esses eventos são despachados a cada poucos milissegundos e também sempre que o mouse se move. O objeto de evento `nativeDragOver` pode ser usado para determinar a posição do mouse sobre o componente. Ter acesso à posição do mouse pode ser útil em situações onde o componente receptor é complexo, mas não é formado de subcomponentes. Por exemplo, se seu aplicativo exibiu um bitmap contendo um mapa de rua e você desejava realçar zonas no mapa quando o usuário arrastasse informações a eles, você poderia usar as coordenadas do mouse reportadas no evento `nativeDragOver` para rastrear a posição do mouse no mapa.

Arrastar e soltar em HTML

Adobe AIR 1.0 e posterior

Para arrastar dados para dentro e fora de um aplicativo baseado em HTML (ou para dentro e fora do HTML exibido em um HTMLLoader), você pode usar eventos de arrastar e soltar HTML. A API de arrastar e soltar HTML permite que você arraste para e de elementos DOM no conteúdo HTML.

Nota: *Você também pode usar as APIs NativeDragEvent e NativeDragManager do AIR ouvindo eventos no objeto HTMLLoader contendo o conteúdo HTML. No entanto, a API HTML é melhor integrada com o DOM HTML e oferece a você controle do comportamento padrão.*

Comportamento padrão de arrastar e soltar

Adobe AIR 1.0 e posterior

O ambiente HTML fornece o comportamento padrão para gestos de arrastar e soltar envolvendo texto, imagens e URLs. Usando o comportamento padrão, você sempre pode arrastar esses tipos de dados para fora de um elemento. No entanto, você pode apenas arrastar texto em um elemento e apenas a elementos em uma região editável de uma página. Quando você arrasta texto entre ou dentro de regiões editáveis de uma página, o comportamento padrão executa uma ação de mover. Quando você arrasta texto para uma região editável de uma região não editável ou de fora do aplicativo, o comportamento padrão executa uma ação de copiar.

Você pode substituir o comportamento padrão manipulando os eventos de arrastar e soltar sozinho. Para cancelar o comportamento padrão, você deve chamar os métodos `preventDefault()` dos objetos despachados para os eventos de arrastar e soltar. Você pode então inserir dados no destino de soltar e remover dados da origem de arrastar conforme necessário para executar a ação escolhida.

Por padrão, o usuário pode selecionar e arrastar qualquer texto e arrastar imagens e links. Você pode usar a propriedade WebKit CSS, `-webkit-user-select`, para controlar como qualquer elemento HTML pode ser selecionado. Por exemplo, se você definir `-webkit-user-select` como `none`, o conteúdo do elemento não podem ser selecionados e, portanto, não podem ser arrastados. Você também pode usar a propriedade CSS `-webkit-user-drag` para controlar se um elemento como um todo pode ser arrastado. No entanto, o conteúdo do elemento é tratado separadamente. O usuário poderia ainda arrastar uma parte selecionada do texto. Para obter mais informações, consulte "CSS no AIR" na página 1009.

Eventos de arrastar e soltar em HTML

Adobe AIR 1.0 e posterior

Os eventos despachados pelo elemento iniciador do qual uma ação de arrastar se origina são:

Evento	Descrição
dragstart	Despachado quando o usuário inicia o gesto de arrastar. O manipulador desse evento pode impedir a ação de arrastar, se necessário, chamando o método preventDefault() do objeto de evento. Para controlar se os dados arrastados podem ser copiados, vinculados ou movidos, defina a propriedade effectAllowed. Texto, imagens e links selecionados são colocados na área de transferência pelo comportamento padrão, mas você pode definir dados diferentes para o gesto de arrastar usando a propriedade dataTransfer do objeto de evento.
arrastar	Despachado continuamente durante o gesto de arrastar.
dragend	Despachado quando o usuário libera o botão do mouse para encerrar o gesto de arrastar.

Os eventos despachados por um destino de arrastar são:

Evento	Descrição
dragover	Despachado continuamente enquanto o gesto de arrastar permanece dentro dos limites do elemento. O manipulador para esse evento deve definir a propriedade dataTransfer.dropEffect para indicar se a ação de soltar resultará em uma ação de copiar, mover ou vincular se o usuário liberar o mouse.
dragenter	Despachado quando o gesto de soltar entrar nos limites de um elemento. Se você alterar alguma propriedade de um objeto dataTransfer em um manipulador de eventos dragenter, essas alterações serão rapidamente substituídas pelo próximo evento dragover. Por outro lado, existe um pequeno atraso entre um evento dragenter e o primeiro dragover que pode fazer com que o cursor pisque se diferentes propriedades forem definidas. Em vários casos, você pode usar o mesmo manipulador de eventos para ambos os eventos.
dragleave	Despachado quando o gesto de arrastar deixa os limites do elemento.
drop	Despachado quando o usuário solta os dados no elemento. Os dados sendo arrastados podem apenas ser acessados dentro do manipulador desse evento.

O objeto de evento despachado em resposta a esses eventos é similar a um evento do mouse. Você pode usar propriedades de evento de mouse como (`clientX`, `clientY`) e (`screenX`, `screenY`), para determinar a posição do mouse.

A propriedade mais importante de um objeto de evento arrastar é `dataTransfer`, que contém os dados que estão sendo arrastados. O objeto `dataTransfer` em si possui as seguintes propriedades e métodos:

Propriedade ou método	Descrição
effectAllowed	O efeito permitido pela origem da ação de arrastar. Normalmente, o manipulador do evento dragstart define esse valor. Consulte "Efeitos de arrastar em HTML" na página 638.
dropEffect	O efeito escolhido pelo destino ou usuário. Se você define `dropEffect` em um manipulador de eventos `dragover` ou `dragenter`, o AIR atualiza o cursor do mouse para indicar o efeito que ocorre se o usuário libera o mouse. Se a definição de `dropEffect` não corresponder a um dos efeitos permitidos, nenhuma ação de soltar será permitida e o cursor *unavailable* será exibido. Se não tiver definido `dropEffect` em resposta a um evento `dragover` ou `dragenter` mais recente, o usuário poderá escolher dentre os efeitos permitidos com as teclas do modificador do sistema operacional padrão. O efeito final é relatado pela propriedade `dropEffect` do objeto despachado para `dragend`. Se o usuário abandonar a ação de soltar liberando o mouse fora de um destino aceitável, `dropEffect` será definido como `none`.
tipos	Uma matriz contendo as sequências de caracteres de tipo MIME para cada formato de dados é apresentada no objeto `dataTransfer`.
getData(mimeType)	Obtém os dados no formato especificado pelo parâmetro `mimeType`. O método `getData()` pode apenas ser chamado em resposta ao evento `drop`.
setData(mimeType)	Adiciona dados a `dataTransfer` no formato especificado pelo parâmetro mimeType. Você pode adicionar dados em vários formatos chamando `setData()` para cada tipo MIME. Qualquer dado colocado no objeto `dataTransfer` pelo comportamento de arrastar padrão é limpo. O método `setData()` pode apenas ser chamado em resposta ao evento `dragstart`.
clearData(mimeType)	Limpa qualquer dado no formato especificado pelo parâmetro `mimeType`.
setDragImage(image, offsetX, offsetY)	Define uma imagem de arrastar personalizada. O método `setDragImage()` só pode ser chamado em resposta ao evento dragstart e somente quando um elemento inteiro de HTML é arrastado ao se definir seu estilo CSS `-webkit-user-drag` ao `element`. O parâmetro `image` pode ser um elemento JavaScript ou um objeto Image.

Tipos MIME para arrastar e soltar HTML

Adobe AIR 1.0 e posterior

Os tipos MIME a usar com o objeto `dataTransfer` de um evento arrastar e soltar HTML incluem:

Formato de dados	Tipo MIME
Texto	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
Lista de arquivos	"application/x-vnd.adobe.air.file-list"

Você também pode usar outras sequências de caracteres MIME, incluindo as definidas por aplicativo. No entanto, outros aplicativos não podem reconhecer ou usar os dados transferidos. É sua responsabilidade adicionar dados ao objeto `dataTransfer` no formato esperado.

Importante: *Apenas código em execução na caixa de proteção do aplicativo pode acessar arquivos soltos. Tentar ler ou definir qualquer propriedade de um objeto File em uma caixa de proteção de não aplicativo gera um erro de segurança. Consulte "Tratamento de arquivos soltos em caixas de proteção HTML de não aplicativo" na página 642 para obter mais informações.*

Efeitos de arrastar em HTML

Adobe AIR 1.0 e posterior

O iniciador do gesto de arrastar pode limitar os efeitos de arrastar permitido definindo a propriedade `dataTransfer.effectAllowed` no manipulador para o evento `dragstart`. Os seguintes valores de sequências de caracteres podem ser usados:

Valor de string	Descrição
"none"	Nenhuma operação de arrastar é permitida.
"copiar"	Os dados serão copiados ao destino, deixando o original no lugar.
"link"	Os dados serão compartilhados com o destino de soltar usando um link de volta ao original.
"mover"	Os dados serão copiados para o destino e removidos do local original.
"copyLink"	Os dados podem ser copiados ou vinculados.
"copyMove"	Os dados podem ser copiados ou movidos.
"linkMove"	Os dados podem ser vinculados ou movidos.
"all"	Os dados podem ser copiados, movidos ou vinculados. *All* é o efeito padrão quando você impede o comportamento padrão.

O destino do gesto de arrastar pode definir a propriedade `dataTransfer.dropEffect` para indicar a ação tomada se o usuário concluir a ação de soltar. Se o efeito de soltar for uma das ações permitidas, o sistema exibe o cursor apropriado de copiar, mover ou vincular. Caso contrário, o sistema exibe o cursor *unavailable*. Se nenhum efeito de soltar for definido pelo destino, o usuário poderá escolher dentre as ações permitidas com as teclas do modificador.

Defina o valor `dropEffect` nos manipuladores para os eventos `dragover` e `dragenter`:

```
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";
}
```

Nota: *Embora você deva sempre definir a propriedade `dropEffect` no manipulador para `dragenter`, esteja ciente que o próximo evento `dragover` redefine a propriedade para seu valor padrão. Defina `dropEffect` em resposta aos dois eventos.*

Arrastar dados para fora de um elemento HTML

Adobe AIR 1.0 e posterior

O comportamento padrão permite que a maior parte do conteúdo em uma página HTML seja copiada pela ação de arrastar. Você pode controlar o conteúdo permitido a ser arrastado usando as propriedades CSS `-webkit-user-select` e `-webkit-user-drag`.

Substitua o comportamento padrão de arrastar para fora no manipulador para o evento `dragstart`. Chame o método `setData()` da propriedade `dataTransfer` do objeto de evento para colocar seus próprios dados no gesto de arrastar.

Para indicar que efeitos de arrastar um objeto de origem suporta quando você não está contando com o comportamento padrão, define a propriedade `dataTransfer.effectAllowed` do objeto de evento despachado para o evento `dragstart`. Você pode escolher qualquer combinação de efeitos. Por exemplo, se um elemento de origem suportar os efeitos *copy* e *link*, defina a propriedade como `"copyLink"`.

Definição dos dados arrastados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Adicione os dados para o gesto de arrastar no manipulador para o evento `dragstart` com a propriedade `dataTransfer`. Use o método `dataTransfer.setData()` para colocar dados na área de transferência, transmitindo o tipo MIME e os dados a transferir.

Por exemplo, se você tinha um elemento de imagem no seu aplicativo, com a id *imageOfGeorge*, você poderia usar o seguinte manipulador de eventos dragstart. Esse exemplo adiciona representações de uma imagem de George em vários formatos de dados, que aumenta a probabilidade de que outros aplicativos possam usar os dados arrastados.

```
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));
}
```

Nota: *Quando você chama o método `setData()` de objeto `dataTransfer`, nenhum dado é adicionado pelo comportamento padrão de arrastar e soltar.*

Arrastar dados para dentro de um elemento HTML

Adobe AIR 1.0 e posterior

O comportamento padrão apenas permite que texto seja arrastado em regiões editáveis da página. Você pode especificar que um elemento e seus filhos podem se tornar editáveis incluindo o atributo `contenteditable` na tag de abertura do elemento. Você também pode tornar um documento inteiro editável definindo a propriedade `designMode` de objeto de documento como `"on"`.

Você pode suportar um comportamento alternativo de arrastar para dentro em uma página manipulando os eventos `dragenter`, `dragover` e `drop` para qualquer elemento que pode aceitar dados arrastados.

Ativação de arrastar para dentro

Adobe AIR 1.0 e posterior

Para manipular o gesto de arrastar para dentro, você deve primeiro cancelar o comportamento padrão. Ouça os eventos `dragenter` e `dragover` em qualquer elemento HTML que deseja usar como destinos de soltar. Nos manipuladores desses eventos, chame o método `preventDefault()` do objeto de evento despachado. Cancelar o comportamento padrão permite que regiões não editáveis recebam uma ação de soltar.

Obtenção de dados soltos

Adobe AIR 1.0 e posterior

Você pode acessar os dados soltos no manipulador para o evento `ondrop`:

```
function doDrop(event){
    droppedText = event.dataTransfer.getData("text/plain");
}
```

Use o método `dataTransfer.getData()` para ler dados na área de transferência, transmitindo o tipo MIME do formato de dados a ler. Você pode descobrir que formatos de dados estão disponíveis usando a propriedade `types` do objeto `dataTransfer`. A matriz `types` contém a sequência de caracteres de tipo MIME de cada formato disponível.

Quando você cancela o comportamento padrão nos eventos dragenter ou dragover, você é responsável por inserir qualquer dado solto em seu próprio lugar no documento. Nenhuma API existe para converter uma posição do mouse em um ponto de inserção dentro de um elemento. Essa limitação pode dificultar a implementação de gestos de arrastar de tipo de inserção.

Exemplo: Substituição do comportamento padrão de arrastar para dentro HTML

Adobe AIR 1.0 e posterior

Esse exemplo implementa um destino de soltar que exibe uma tabela mostrando cada formato de dados disponível no item solto.

O comportamento padrão é usado para permitir que texto, links e imagens sejam arrastados no aplicativo. O exemplo substitui o comportamento padrão de arrastar para dentro para o elemento div que serve como o destino de soltar. A etapa principal para habilitar conteúdo não editável para aceitar um gesto de arrastar para dentro é chamar o método `preventDefault()` do objeto de evento despachado para os eventos `dragenter` e `dragover`. Em resposta a um evento `drop`, o manipulador converte os dados transferidos em um elemento de linha HTML e insere a linha em uma tabela para exibição.

```
 <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>
```

Tratamento de arquivos soltos em caixas de proteção HTML de não aplicativo

Adobe AIR 1.0 e posterior

O conteúdo de não aplicativo não pode acessar os objetos File resultantes quando arquivos são arrastados em um aplicativo do AIR. Nem é possível transmitir um desses objetos File para conteúdo de aplicativo por uma ponte de caixa de proteção. (As propriedades do objeto devem ser acessadas durante a serialização.) No entanto, você pode ainda soltar arquivos no seu aplicativo ouvindo os eventos nativeDragDrop do AIR no objeto HTMLLoader.

Normalmente, se um usuário solta um arquivo em um quadro que hospeda conteúdo de não aplicativo, o evento de soltar não propaga do filho ao pai. No entanto, como os eventos despachados pelo HTMLLoader (que é o contêiner para todo o conteúdo HTML em um aplicativo do AIR) não são parte do fluxo de evento HTML, você pode ainda receber o evento de soltar em conteúdo de aplicativo.

Para receber o evento para um arquivo solto, o documento pai adiciona um ouvinte de evento ao objeto HTMLLoader usando a referência fornecida por `window.htmlLoader`:

```
 window.htmlLoader.addEventListener("nativeDragDrop",function(event){
    var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
    air.trace(filelist[0].url);
});
```

O exemplo a seguir usa um documento pai que carrega uma página filha em uma caixa de proteção remota (http://localhost/). O pai ouve o evento `nativeDragDrop` no objeto HTMLLoader e marca a URL do arquivo.

```
<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>
```

O documento filho deve apresentar um destino de soltar válido chamando o método `preventDefault()` do objeto Event nos manipuladores de eventos `dragenter` e `dragover`. Do contrário, o evento de soltar pode nunca ocorrer.

```
<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>
```

Soltando promessas de arquivo

Adobe AIR 2 e posterior

Uma promessa de arquivo é um formato de área de transferência de arrastar e soltar que permite que um usuário arraste um arquivo que ainda não existe para fora de um aplicativo do AIR. Por exemplo, usando as promessas de arquivo, seu aplicativo poderia permitir que um usuário arraste um ícone de proxy para uma pasta do computador. O ícone de proxy representa um arquivo ou alguns dados conhecidos que estão disponíveis em um URL. Depois que o usuário solta o ícone, o tempo de execução baixa os dados e escreve o arquivo no local em que foi solto.

Você pode usar a classe URLFilePromise em um aplicativo do AIR para arrastar e soltar arquivos acessíveis em um URL. A implementação do URLFilePromise é fornecido na biblioteca do aircore como parte do SDK do AIR 2. Utilize o arquivo aircore.swc ou o aircore.swf encontrados no diretório frameworks/libs/air.

Alternativamente, você pode implementar sua própria lógica de promessa de arquivo utilizando a interface IFilePromise (que é definida no pacote flash.desktop do tempo de execução).

As promessas de arquivo são semelhantes em conceito à renderização adiada usando uma função de tratamento de dados na área de transferência. Use as promessas de arquivo em vez da renderização adiada ao arrastar e soltar os arquivos. A técnica de renderização adiada pode levar a pausas indesejáveis no gesto de arrastar à medida que os dados são gerados ou baixados. Use a renderização adiada para as operações de copiar e colar (para as quais as promessas do arquivo não recebem suporte).

Limitações ao usar as promessas de arquivo

As promessas de arquivo têm as seguintes limitações em comparação com outros formatos de dados que você pode colocar em uma área de transferência de arrastar e soltar:

- As promessas de arquivo só podem ser arrastadas para fora de um aplicativo do AIR, não podendo ser soltas em um arquivo AIR.
- As promessas de arquivo não recebem suporte em todos os sistemas operacionais. Use a propriedade `Clipboard.supportsFilePromise` para testar se as promessas de arquivo recebem suporte no sistema de host. Nos sistemas que não oferecem suporte às promessas de arquivo, forneça um mecanismo alternativo para baixar ou gerar os dados do arquivo.
- As promessas de arquivo não podem ser usadas com a área de transferência de copiar e colar (`Clipboard.generalClipboard`).

Mais tópicos da Ajuda

flash.desktop.IFilePromise

air.desktop.URLFilePromise

Soltando arquivos remotos

Adobe AIR 2 e posterior

Use a classe URLFilePromise para criar objetos de promessa de arquivo que representam arquivos ou dados disponíveis em um URL. Acrescente um ou mais objetos de promessa de arquivo à área de transferência usando o formato `FILE_PROMISE_LIST` da área. No exemplo a seguir, um arquivo único, disponível em http://www.example.com/foo.txt, é baixado e salvo no local em que será solto como bar.txt. (Os nomes do arquivo remoto e local não precisam coincidir.)

```
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 );
}
```

Você pode permitir que o usuário arraste mais de um arquivo por vez acrescentando mais objetos de promessa de arquivo à matriz atribuída à área de transferência. Também é possível especificar os subdiretórios na propriedade `relativePath`, de forma que alguns ou todos os arquivos incluídos na operação sejam colocados em uma subpasta relativa ao local em que ele será solto.

O exemplo a seguir ilustra como iniciar uma operação de soltar que inclui múltiplas promessas de arquivo. Neste exemplo, uma página html, *article.html*, é colocada na área de transferência como promessa de arquivo, juntamente com seus dois arquivos de imagem vinculados. As imagens são copiadas para uma subpasta *images*, de forma que os links relativos sejam mantidos.

```
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 );
}
```

Implementação da interface IFilePromise

Adobe AIR 2 e posterior

Para fornecer promessas de arquivo para os recursos que não podem ser acessados com um objeto URLFilePromise, você pode implementar a interface IFilePromise em uma classe personalizada. A interface IFilePromise define os métodos e propriedades usados pelo tempo de execução do AIR para acessar os dados a serem gravados em um arquivo depois que a promessa é solta.

Uma implementação IFilePromise passa outro objeto ao tempo de execução do AIR que fornece os dados para a promessa de arquivo. Esse objeto deve implementar a interface IDataInput, que o tempo de execução do AIR usa para ler os dados. Por exemplo, a classe URLFilePromise, que implementa IFilePromise, usa um objeto URLStream como fornecedor dos dados.

O AIR pode ler os dados de forma síncrona ou assíncrona. A implementação IFilePromise informa qual modo de acesso é suportado ao devolver o valor apropriado na propriedade `isAsync`. Se for fornecido acesso aos dados assíncronos, o objeto do fornecedor de dados deve implementar a interface IEventDispatcher e despachar os eventos necessários, como `open`, `progress` e `complete`.

Você pode usar uma classe personalizada ou uma das classes incorporadas seguintes, como fornecedor de dados para uma promessa de dados:

- ByteArray (síncrono)

- FileStream (síncrono ou assíncrono)
- Socket (assíncrono)
- URLStream (assíncrono)

Para implementar a interface IFilePromise, você deve fornecer o código para as funções e propriedades seguintes:

- `open():IDataInput` — Devolve o objeto do fornecedor de dados dos quais os dados para o arquivo prometido são lidos. O objeto deve implementar a interface IDataInput. Se os dados são fornecidos de forma assíncrona, o objeto também deve implementar a interface IEventDispatcher e despachar os eventos necessários (consulte "Uso do fornecedor de dados assíncrono em uma promessa de arquivo" na página 648).
- `get relativePath():String` — Fornece o caminho, incluindo nome de arquivo, para o arquivo criado. O caminho é resolvido em relação ao local de soltar escolhido pelo usuário na operação de arrastar e soltar. Para ter certeza de que o caminho usa o caractere separador adequado para o sistema operacional de host, use a constante `File.separator` ao especificar os caminhos que contêm diretórios. Você pode adicionar uma função setter ou usar um parâmetro constructor para permitir que o caminho seja definido no tempo de execução.
- `get isAsync():Boolean` — Informa o tempo de execução do AIR se o objeto do fornecedor de dados fornece seus dados de forma síncrona ou assíncrona.
- `close():void` — Chamado pelo tempo de execução quando os dados são inteiramente lidos (ou um erro impede novas leituras). Você pode usar essa função para os recursos de limpeza.
- `reportError( e:ErrorEvent ):void` — Chamando pelo tempo de execução quando ocorre um erro ao ler os dados.

Todos os métodos IFilePromise são chamados pelo tempo de execução durante uma operação de arrastar e soltar envolvendo a promessa de arquivo. Em geral, sua lógica de aplicativo não deve chamar nenhum desses métodos diretamente.

Uso do fornecedor de dados síncrono em uma promessa de arquivo

Adobe AIR 2 e posterior

A forma mais simples de implementar a interface IFilePromise é usar um objeto de fornecedor de dados síncrono, como ByteArray ou um FileStream síncrono. No exemplo a seguir, é criado um objeto ByteArray, preenchido com dados e retornado quando se chama o método `open()`.

```
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 );
        }
    }
}
```

Na prática, as promessas de arquivo síncronas têm utilidade limitada. Se a quantidade de dados for pequena, você poderia facilmente criar um arquivo em um diretório temporário e acrescentar uma matriz normal da lista de arquivos à área de transferência de arrastar e soltar. Por outro lado, se a quantidade de dados for grande ou for caro gerar os dados de forma computacional, é necessário um processo síncrono longo. Os processos síncronos longos podem bloquear as atualizações de UI por um tempo notável e fazer com que seu aplicativo pareça não responder. Para evitar este problema, você pode criar um fornecedor de dados assíncronos orientado por um cronômetro.

Uso do fornecedor de dados assíncrono em uma promessa de arquivo

Adobe AIR 2 e posterior

Ao usar um objeto de fornecedor de dados assíncrono, a propriedade`isAsync` do IFilePromise deve ser `true` e o objeto devolvido pelo método `open()` deve implementar a interface IEventDispatcher. O tempo de execução ouve os vários eventos alternativos de forma que se possam usar objetos incorporados diferentes como fornecedor de dados. Por exemplo, os eventos `progress` são despachados pelos objetos FileStream e URLStream, ao passo que os eventos `socketData` são despachados pelos objetos Socket. O tempo de execução ouve os eventos apropriados de todos esses objetos.

Os eventos a seguir orientam o processo de leitura de dados do objeto do fornecedor de dados:

- Event.OPEN — Informa ao tempo de execução que a fonte de dados está pronta.
- ProgressEvent.PROGRESS — Informa ao tempo de execução que os dados estão disponíveis. O tempo de execução vai ler a quantidade de dados disponíveis do objeto do fornecedor de dados.
- ProgressEvent.SOCKET_DATA — Informa ao tempo de execução que os dados estão disponíveis. O evento `socketData` é despachado pelos objetos baseados em soquete. Para os outros tipos de objeto, despache um evento `progress`. (O tempo de execução ouve os dois eventos para detectar quando os dados podem ser lidos.)
- Event.COMPLETE — Informa ao tempo de execução que os dados foram todos lidos.
- Event.CLOSE — Informa ao tempo de execução que os dados foram todos lidos. (O tempo de execução ouve `close` e `complete` para essa finalidade.)
- IOErrorEvent.IOERROR — Informa ao tempo de execução que ocorreu um erro ao ler os dados. O tempo de execução aborta a criação do arquivo e chama o método `close()` do IFilePromise.
- SecurityErrorEvent.SECURITY_ERROR — Informa ao tempo de execução que ocorreu um erro de segurança. O tempo de execução aborta a criação do arquivo e chama o método `close()` do IFilePromise.
- HTTPStatusEvent.HTTP_STATUS — Usado, junto com `httpResponseStatus`, pelo tempo de execução para garantir que os dados disponíveis representem o conteúdo desejado, em vez de uma mensagem de erro (como uma página 404). Os objetos baseados no protocolo HTTP devem despachar esse evento.
- HTTPStatusEvent.HTTP_RESPONSE_STATUS — Usado, junto com `httpStatus`, pelo tempo de execução para garantir que os dados disponíveis representam o conteúdo desejado. Os objetos baseados no protocolo HTTP devem despachar esse evento.

O fornecedor de dados deve despachar esses eventos na sequência a seguir:

1. evento `open`
2. Eventos `progress` ou `socketData`
3. Evento `complete` ou `close`

Nota: *Os objetos incorporados, FileStream, Socket e URLStream, despacham os eventos apropriados automaticamente.*

O exemplo a seguir cria uma promessa de arquivo usando um fornecedor de dados personalizado e assíncrono. A classe do fornecedor de dados estende ByteArray (para o suporte do IDataInput) e implementa a interface IEventDispatcher. No evento de cada cronômetro, o objeto gera um pouco de dados e despacha um evento progress para informar ao tempo de execução que os dados estão disponíveis. Quando forem produzidos dados suficientes, o objeto despacha um evento completo.

```
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 );
    }
}
}
```

Nota: *Como a classe AsyncDataProvider no exemplo estende ByteArray, ela não pode estender também o EventDispatcher. Para implementar a interface IEventDispatcher, a classe usa um objeto EventDispatcher interno e encaminha as chamadas do método IEventDispatcher para aquele objeto interno. Você também poderia estender o EventDispatcher e implementar IDataInput (ou implementar as duas interfaces).*

A implementação assíncrona de IFilePromise é quase idêntica à implementação síncrona. As principais diferenças são que `isAsync` retorna `true` e que o método `open()` retorna um objeto de dados assíncrono:

```
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 );
        }
    }
}
```

Capítulo 36: Trabalho com menus

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use as classes na API de menu de contexto para modificar o menu de contexto em aplicativos do Flex e Flash Player com base na Web.

Use as classes na API de menu nativo para definir aplicativo, janela, contexto e menus pop-up no Adobe® AIR®.

Noções básicas do menu

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para ver uma rápida explicação e exemplos de código da criação de menus originais em aplicativos AIR, consulte os seguintes artigos de início rápido no Adobe Developer Connection:

- Adicionando menus nativos em aplicativos AIR (Flex)
- Adicionando menus nativos em aplicativos AIR (Flash)

As classes de menu nativo permitem acessar os recursos do menu nativo do sistema operacional em que o aplicativo está sendo executado. Objetos NativeMenu podem ser usados em menus de aplicativo (disponíveis no Mac OS X), menus de janela (disponíveis no Windows e no Linux), menus de contexto e menus pop-up.

Fora do AIR, você pode usar as classes de menu de contexto para modificar o menu de contexto que o Flash Player automaticamente exibe quando um usuário clica com o botão direito ou clica com a tecla cmd em um objeto em um aplicativo. (O menu de contexto automático não é exibido para os aplicativos AIR.)

Classes de menu

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As classes de menu incluem:

Pacote	Classes
flash.display	• NativeMenu • NativeMenuItem
flash.ui	• ContextMenu • ContextMenuItem
flash.events	• Event • ContextMenuEvent

Variedades de menu

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O AIR oferece suporte aos seguintes tipos de menus:

Menus de contexto Os menus de contexto são exibidos em resposta ao clique com o botão direito do mouse ou ao clique com a tecla Command pressionada em um objeto interativo no conteúdo SWF ou um elemento de documento no conteúdo HTML.

No tempo de execução do Flash Player, um menu de contexto é automaticamente exibido. Você pode usar as classes ContextMenu e ContextMenuItem para adicionar seus próprios comandos ao menu. Também é possível remover alguns comandos incorporados, mas não todos.

No tempo de execução do AIR, você pode criar um menu de contexto usando a classe NativeMenu ou ContextMenu. No conteúdo HTML do AIR, você pode usar as APIs de kit da Web HTML e JavaScript para adicionar menus de contexto a elementos HTML.

Menus de aplicativo (somente AIR) O menu de aplicativo é um menu global aplicável ao aplicativo inteiro. Há suporte para menus de aplicativo no Mac OS X, mas não no Windows nem no Linux. No Mac OS X , o sistema operacional cria automaticamente o menu de aplicativo. Você pode usar a API de menu AIR para adicionar itens e submenus aos menus padrão. Você pode adicionar ouvintes para tratar os comandos de menu existentes. Ou você pode remover itens existentes.

Menus de janela (somente AIR) O menu de janela está associado a uma única janela e é exibido abaixo da barra de título. Os menus podem ser adicionados a uma janela, criando um objeto NativeMenu e atribuindo-o à propriedade `menu` do objeto NativeWindow. Há suporte para menus de janela nos sistemas operacionais Windows e Linux, mas não no Mac OS X. Menus de janela nativos só podem ser usados em janelas com o cromo do sistema.

Menus do ícone de bandeja de encaixe e sistema (somente AIR) Esses menus de ícone são semelhantes aos menus de contexto e são atribuídos a um ícone de aplicativo no encaixe do Mac OS X ou nas áreas de notificação do Windows ou do Linux, na barra de tarefas. Menus de ícone da bandeja do sistema e de encaixe usam a classe NativeMenu. No Mac OS X, os itens no menu são adicionados acima dos itens padrão do sistema operacional. Não há menu padrão no Windows nem no Linux.

Menus pop-up (somente AIR) O menu pop-up AIR é semelhante ao menu de contexto, mas não está necessariamente associado a um objeto de aplicativo ou componente específico. Os menus pop-up podem ser exibidos em qualquer lugar na janela, chamando o método `display()` de qualquer objeto NativeMenu.

Menus personalizados Menus nativos são inteiramente desenhados pelo sistema operacional e, como tal, estão presentes fora dos modelos de processamento Flash e HTML. Em vez de usar menus nativos, você sempre pode criar seus próprios menus não nativos personalizados usando MXML, ActionScript ou JavaScript (no AIR). Esses menus devem ser inteiramente renderizados no conteúdo do aplicativo.

menus do Flex A estrutura do Adobe® Flex™ oferece um conjunto de componentes de menu do Flex. Os menus do Flex são desenhados pelo tempo de execução, e não pelo sistema operacional, e não são menus *nativos*. O componente de menu do Flex pode ser usado em janelas do Flex que não têm o cromo do sistema. Outra vantagem de usar o componente do menu Flex é que você pode especificar de forma clara os menus no formato MXML. Se estiver usando a estrutura do Flex, use as classes de menu do Flex em menus de janela, em vez das classes nativas.

Menus padrão (apenas AIR)

Os menus padrão a seguir são fornecidos pelo sistema operacional ou uma classe incorporada do AIR:

- Menu de aplicativo no Mac OS X
- Menu de ícone de encaixe no Mac OS X

- Menu de contexto de textos e imagens selecionados em conteúdo HTML
- Menu de contexto de texto selecionado no objeto TextField (ou em um objeto que estende o TextField)

Sobre menus de contexto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No conteúdo SWF, qualquer objeto herdado de InteractiveObject pode ser determinado um menu de contexto atribuindo um objeto de menu à respectiva propriedade `contextMenu`. Vários comandos são incluídos por padrão, como Avançar, Voltar, Imprimir, Qualidade e Zoom. No tempo de execução do AIR, o objeto de menu atribuído ao `contextMenu` pode ser do tipo NativeMenu ou do tipo ContextMenu. No tempo de execução do Flash Player, apenas a classe ContextMenu está disponível.

Você pode ouvir eventos do menu nativo ou de menus contextuais ao usar as classes ContextMenu e ContextMenuItem; as duas são despachadas. Um benefício fornecido pelas propriedades do objeto ContextMenuEvent é que contextMenuOwner identifica o objeto ao qual o menu está associado e `mouseTarget` identifica o objeto que foi clicado para abrir o menu. Essas informações não estão disponíveis do objeto NativeMenuEvent.

O exemplo a seguir cria um Sprite e adiciona um menu de contexto de edição simples:

```
 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");}
```

Nota: *Ao contrário do conteúdo SWF exibido em um ambiente de navegador, os menus de contexto no AIR não têm nenhum comando incorporado.*

Personalização de um menu de contexto do Flash Player

Em um navegador ou projetor, os menus de contexto no conteúdo SWF sempre têm itens incorporados. Você pode remover todos esses comandos padrão do menu, exceto Configurações e Sobre. Configurar a propriedade `showDefaultContextMenu` do palco como `false` remove esses comandos do menu de contexto.

Para criar um menu de contexto personalizado para um objeto de exibição específico, crie uma nova ocorrência da classe ContextMenu, chame o método `hideBuiltInItems()` e atribua essa ocorrência à propriedade `contextMenu` dessa instância DisplayObject. O exemplo a seguir fornece um quadrado desenhado dinamicamente com um comando de menu de contexto para alterá-lo para uma cor aleatória:

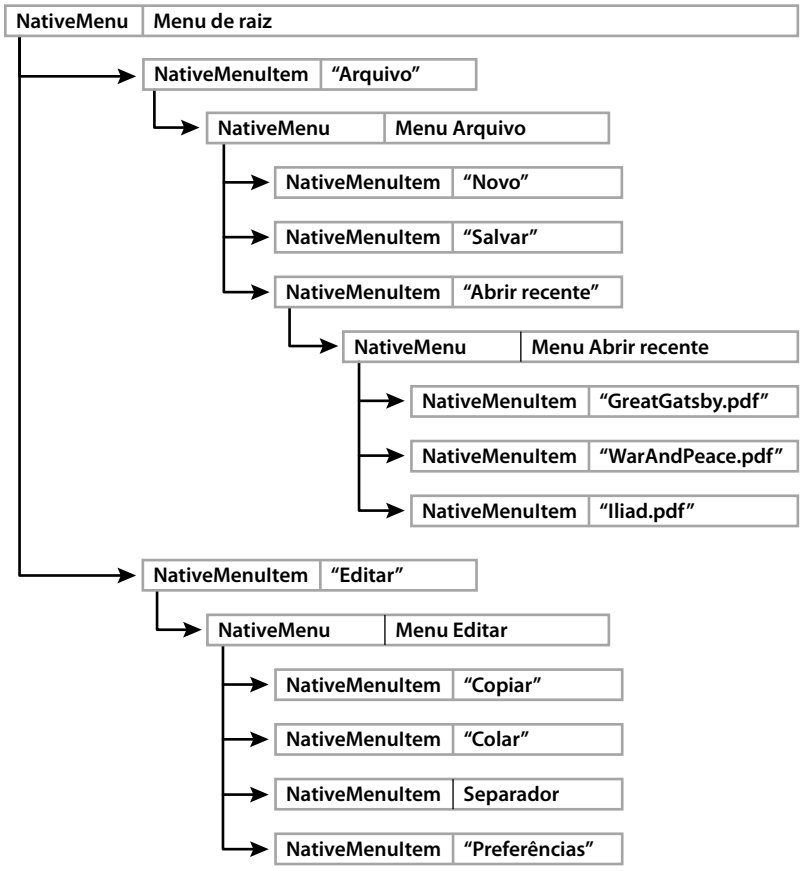
```
var square:Sprite = new Sprite();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.x =
square.y = 10;
addChild(square);

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);
}
```

Estrutura do menu nativo (AIR)

Adobe AIR 1.0 e posterior

Os menus nativos são de natureza hierárquica. Os objetos NativeMenu contêm objetos-filho NativeMenuItem. Os objetos NativeMenuItem que representam submenus, por sua vez, podem conter objetos NativeMenu. O objeto de menu de nível superior ou raiz na estrutura representa a barra de menu de menus de aplicativo e janela. (Menus de contexto, ícone e pop-up não têm uma barra de menu).

O diagrama a seguir ilustra a estrutura de um menu típico. O menu raiz representa a barra de menu e contém dois itens de menu que fazem referência ao submenu *File* e ao submenu *Edit*. O submenu Arquivo nessa estrutura contém dois itens de comando e um item que faz referência ao submenu *Open Recent Menu*, que contém, ele mesmo, três itens. O submenu Edit contém três comandos e um separador.



Definir um submenu requer um objeto NativeMenu e um objeto NativeMenuItem. O objeto NativeMenuItem define o rótulo exibido no menu pai e permite ao usuário abrir o submenu. O objeto NativeMenu funciona como um recipiente de itens no submenu. O objeto NativeMenuItem se refere ao objeto NativeMenu pela propriedade `submenu` NativeMenuItem.

Para exibir um exemplo de código que cria esse menu, consulte "Exemplo de menu nativo: menu de janela e de aplicativo (AIR)" na página 664.

Eventos menu

Adobe AIR 1.0 e posterior

Os objetos NativeMenu e NativeMenuItem despacham os eventos `preparing`, `displaying` e `select`:

Preparing: sempre que o objeto está prestes a começar uma interação com o usuário, o menu e seus itens de menu despacham um evento `preparing` para qualquer ouvinte registrado. A interação inclui abrir o menu ou selecionar um item com um atalho no teclado.

Nota: *O evento* `preparing` *fica disponível somente no Adobe AIR 2.6 e posteriores.*

Displaying: Imediatamente antes de o menu ser exibido, o menu e os respectivos itens de menu despacham o evento `displaying` para qualquer ouvinte registrado.

Os eventos `preparing` e `displaying` dão a oportunidade de atualizar o conteúdo do menu ou a aparência do item antes de eles serem mostrados ao usuário. Por exemplo, no ouvinte do evento `displaying` do menu "Abrir recentes", você pode alterar os itens de menu para que reflitam a lista atual de documentos exibidos recentemente.

Se você remover o item de menu cujo atalho no teclado acionou o evento `preparing`, a interação com o menu é efetivamente cancelada e o evento `select` não é despachado.

As propriedades `target` e `currentTarget` do evento são ambas um objeto em que o ouvinte está registrado: seja o próprio menu ou um de seus itens.

O evento `preparing` é despachado antes do evento `displaying`. Geralmente, você ouve um dos dois eventos, mas não ambos.

Select: Quando um item de comando é escolhido pelo usuário, o item despacha um evento `select` para qualquer ouvinte registrado. Itens de submenu e separadores não podem ser selecionados, portanto, nunca despacham um evento `select`.

O evento `select` surge em bolha de um item de menu para o menu que o contém, em cima do menu raiz. Você pode ouvir eventos `select` diretamente em um item e pode ouvir mais acima na estrutura de menu. Quando você ouvir o evento `select` em um menu, poderá identificar o item selecionado usando a propriedade `target` do evento. Conforme o evento surge em bolha pela hierarquia de menu, a propriedade `currentTarget` do objeto event identifica o objeto de menu atual.

Nota: *Os objetos ContextMenu e ContextMenuItem despacham eventos* `menuItemSelect` *e* `menuSelect`*, bem como eventos* `select`*,* `preparing` *e* `displaying`*.*

Equivalentes de chave para os comandos do menu nativo (AIR)

Adobe AIR 1.0 e posterior

Você pode atribuir um equivalente de tecla (às vezes chamado de acelerador) em um comando de menu. O item do menu despacha o evento `select` para qualquer ouvinte registrado quando a tecla ou combinação de teclas é pressionada. O menu que contém o item deve ser parte do menu do aplicativo ou a janela ativa do comando que deve ser chamado.

Os equivalentes de tecla têm duas partes, uma sequência representando a tecla primária e uma matriz de teclas do modificador, que também devem ser pressionadas. Para atribuir a tecla primária, defina a propriedade `keyEquivalent` do item de menu para a sequência de caractere único dessa tecla. Se você usar uma letra maiúscula, a tecla Shift é adicionada à matriz do modificador automaticamente.

No Mac OS X, o modificador padrão é a tecla Command (`Keyboard.COMMAND`). No Windows e no Linux, é a tecla Control (`Keyboard.CONTROL`). Essas teclas padrão são adicionadas automaticamente à matriz do modificador. Para atribuir teclas do modificador diferentes, atribua uma nova matriz contendo os códigos de tecla desejados à propriedade `keyEquivalentModifiers`. A matriz padrão é sobrescrita. Se você usar os modificadores padrão ou atribuir a sua própria matriz de modificador, a tecla Shift será adicionada, se a sequência atribuída à propriedade `keyEquivalent` for uma letra maiúscula. As constantes dos códigos de tecla para usar nas teclas do modificador são definidas na classe Keyboard.

A sequência de equivalente de tecla atribuída é exibida automaticamente ao lado no nome do item de menu. O formato depende do sistema operacional do usuário e das preferências do sistema.

Nota: *Se você atribuir o valor* `Keyboard.COMMAND` *a uma matriz de modificador de tecla no sistema operacional do Windows, nenhum equivalente de tecla será exibido no menu. No entanto, a tecla Control deve ser usada para ativar o comando de menu.*

O exemplo a seguir atribui `Ctrl+Shift+G` como equivalente de tecla de um item de menu:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
```

Este exemplo atribui `Ctrl+Shift+G` como equivalente de tecla, definindo diretamente a matriz do modificador:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
item.keyEquivalentModifiers = [Keyboard.CONTROL];
```

Nota: *Os equivalentes de tecla só são ativados em menus de aplicativo e de janela. Se você adicionar um equivalente de tecla a um menu de contexto ou de pop-up, o equivalente de tecla é exibido no rótulo do menu, mas o comando de menu associado nunca será chamado.*

Mnemônicos (AIR)

Adobe AIR 1.0 e posterior

Os mnemônicos são parte da interface de teclado do sistema operacional em menus. Linux, Mac OS X e Windows permitem a usuários abrir menus e selecionar comandos com o teclado, mas há diferenças sutis.

No Mac OS X, o usuário digita a primeira ou as duas primeiras letras do menu ou comando e, em seguida, pressiona a tecla retornar. A propriedade `mnemonicIndex` é ignorada.

No Windows, apenas uma única letra é significativa. Por padrão, a letra significativa é o primeiro caractere no rótulo, mas se você atribuir um mnemônico ao item de menu, o caractere significativo se torna a letra designada. Se dois itens em um menu têm o mesmo caractere significativo (quer um mnemônico tenha ou não sido atribuído), a interação de teclado do usuário com o menu muda um pouco. Em vez de pressionar uma única letra para selecionar o menu ou comando, o usuário deve pressionar a letra quantas vezes for necessário para realçar o item desejado e, em seguida, pressionar a tecla Enter para concluir a seleção. Para manter um comportamento consistente, você deve atribuir um mnemônico exclusivo a cada item no menu, em menus de janela.

No Linux, nenhum mnemônico padrão é fornecido. Você deve especificar um valor para a propriedade `mnemonicIndex` de um item de menu para fornecer um mnemônico.

Especifique o caractere mnemônico como índice na sequência de rótulo. O índice do primeiro caractere em um rótulo é 0. Portanto, para usar "r" como o mnemônico de um item de menu rotulado, "Format", você deve definir a propriedade `mnemonicIndex` igual a 2.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.mnemonicIndex = 2;
```

Estado do item de menu

Adobe AIR 1.0 e posterior

Os itens de menu têm duas propriedades de estados, `marcados` e `ativados`:

marcado Defina como `true` para exibir uma marca de seleção ao lado do rótulo de item.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.checked = true;
```

ativado Alterne o valor entre `true` e `false` para controlar se o comando está ativado. Itens desativados ficam visualmente “esmaecidos” e não despacham eventos `select`.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.enabled = false;
```

Anexar objeto a um item de menu

Adobe AIR 1.0 e posterior

A propriedade `data` da classe NativeMenuItem permite fazer referência a um objeto arbitrário em cada item. Por exemplo, no menu "Open Recent", você pode atribuir o objeto File de cada documento a cada item do menu.

```
var file:File = File.applicationStorageDirectory.resolvePath("GreatGatsby.pdf")
var menuItem:NativeMenuItem = docMenu.addItem(new NativeMenuItem(file.name));
menuItem.data = file;
```

Criação de menus nativos (AIR)

Adobe AIR 1.0 e posterior

Este tópico descreve como criar os vários tipos de menu nativos suportados pelo AIR.

Criação de objeto do menu raiz

Adobe AIR 1.0 e posterior

Para criar um objeto NativeMenu para servir como raiz do menu, use o construtor NativeMenu:

```
var root:NativeMenu = new NativeMenu();
```

Em menus de aplicativo e janela, o menu raiz representa a barra de menu e deve conter apenas itens que abram submenus. Menus de contexto e pop-up não têm barra de menu, portanto, o menu raiz pode conter comandos e linhas separadoras, bem como submenus.

Depois que o menu é criado, você pode adicionar itens de menu. Os itens aparecem no menu na ordem em que são adicionados, a menos que você os adicione em um índice específico, usando o método `addItemAt()` de um objeto de menu.

Atribua o menu como aplicativo, janela, ícone ou menu de contexto ou exiba-o como menu pop-up, conforme mostrado nas seções a seguir:

Configuração do menu de aplicativo ou menu de janela

É importante que o código possua suporte a ambos menus do aplicativo (suporte no Mac OS) e menus de janela (suporte em outros sistems operacionais)

```
var root:NativeMenu = new NativeMenu();
if (NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu = root;
}
else if (NativeWindow.supportsMenu)
{
    nativeWindow.menu = root;
}
```

Nota: *O Mac OS define um menu contendo itens padrão para cada aplicativo. Atribuir um novo objeto NativeMenu à propriedade* `menu` *do objeto NativeApplication substitui o menu padrão. Você também pode usar o menu padrão, em vez de substitui-lo.*

O Adobe Flex fornece uma classe FlexNativeMenu para criar facilmente menus que funcionam em várias plataformas. Se estiver usando a estrutura do Flex, use as classes FlexNativeMenu no lugar da classe NativeMenu.

Configuração de menu de contexto em um objeto interativo

```
interactiveObject.contextMenu = root;
```

Configuração de menu de ícone de encaixe ou o menu de ícone da bandeja do sistema

É importante que o código possua suporte a ambos menus do aplicativo (suporte no Mac OS) e menus de janela (suporte em outros sistems operacionais)

```
if (NativeApplication.supportsSystemTrayIcon)
{
    SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
}
else if (NativeApplication.supportsDockIcon)
{
    DockIcon(NativeApplication.nativeApplication.icon).menu = root;
}
```

Nota: *O Mac OS X define um menu padrão para o ícone de encaixe do aplicativo. Ao atribuir um novo NativeMenu à propriedade de menu do objeto DockIcon, os itens nesse menu são exibidos acima dos itens padrão. Você não pode remover, acessar nem modificar os itens de menu padrão.*

Exibição de menu como pop-up

```
root.display(stage, x, y);
```

Mais tópicos da Ajuda

Desenvolvimento de aplicativos AIR para várias plataformas

Criação de submenu

Adobe AIR 1.0 e posterior

Para criar um submenu, você adiciona o objeto NativeMenuItem ao menu pai e, em seguida, atribui o objeto NativeMenu, definindo o submenu para a propriedade `submenu` do item. O AIR oferece duas maneiras de criar itens de submenu e respectivos objetos de menu associados:

Você pode criar um item de menu e respectivo objeto de menu relacionado em uma etapa, com o método `addSubmenu()`:

```
var editMenuItem:NativeMenuItem = root.addSubmenu(new NativeMenu(), "Edit");
```

Você também pode criar o item de menu e atribuir separadamente o objeto de menu à respectiva propriedade `submenu`:

```
var editMenuItem:NativeMenuItem = root.addItem("Edit", false);
editMenuItem.submenu = new NativeMenu();
```

Criação de comando de menu

Adobe AIR 1.0 e posterior

Para criar um comando de menu, adicione o objeto NativeMenuItem a um menu e adicione um ouvinte de evento que faça referência à função que implementa o comando de menu:

```
var copy:NativeMenuItem = new NativeMenuItem("Copy", false);
copy.addEventListener(Event.SELECT, onCopyCommand);
editMenu.addItem(copy);
```

Você pode ouvir o evento `select` no próprio item de comando (conforme mostrado no exemplo) ou pode ouvir o evento `select` em um objeto de menu pai.

Nota: *Os itens de menu que representam submenus e linhas separadoras não despacham eventos `select` e, portanto, não podem ser usados como comandos.*

Criação de linha separadora de menu

Adobe AIR 1.0 e posterior

Para criar uma linha separadora, crie um NativeMenuItem, definindo o parâmetro `isSeparator` como `true` no construtor. Em seguida, adicione o item separador ao menu no local correto:

```
var separatorA:NativeMenuItem = new NativeMenuItem("A", true);
editMenu.addItem(separatorA);
```

O rótulo especificado para o separador, se houver, não é exibido.

Sobre menus de contexto em HTML (AIR)

Adobe AIR 1.0 e posterior

No conteúdo HTML exibido com o objeto HTMLLoader, o evento `contextmenu` pode ser usado para exibir um menu de contexto. Por padrão, o menu de contexto é exibido automaticamente quando o usuário chama o evento de menu de contexto no texto selecionado (clicando com o botão direito do mouse ou clicando com a tecla Command pressionada no texto). Para evitar que o menu padrão seja aberto, ouça o evento `contextmenu` e chame o método `preventDefault()` do objeto de evento:

```
function showContextMenu(event){
    event.preventDefault();
}
```

Em seguida, você pode exibir um menu de contexto personalizado usando técnicas DHTML ou exibindo o menu de contexto nativo do AIR. O exemplo a seguir exibe um menu de contexto nativo, chamando o método `display()` do menu em resposta ao evento `contextmenu` de HTML:

```
<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>
```

Exibição de menus nativos pop-up (AIR)

Adobe AIR 1.0 e posterior

Você pode exibir qualquer objeto NativeMenu em um tempo e local arbitrários acima de uma janela, chamando o método `display()` do menu. O método requer uma referência ao palco, portanto, apenas o conteúdo na caixa de proteção do aplicativo pode exibir o menu como pop-up.

O método a seguir exibe o menu definido por um objeto NativeMenu chamado `popupMenu` em resposta a um clique do mouse:

```
private function onMouseClick(event:MouseEvent):void {
    popupMenu.display(event.target.stage, event.stageX, event.stageY);
}
```

Nota: *O menu não precisa ser exibido em resposta direta ao evento. Qualquer método pode chamar a função* `display()`.

Tratamento de itens de menu

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O menu despacha eventos quando o usuário o seleciona ou quando seleciona um item de menu.

Resumo de eventos de classes de menu

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Adicione ouvintes de eventos a menus ou itens individuais para tratar eventos de menu.

Objeto	Eventos despachados
NativeMenu (AIR)	Event.PREPARING (Adobe AIR 2.6 e posterior) Event.DISPLAYING Event.SELECT (propagado de itens filho e submenus)
NativeMenuItem (AIR)	Event.PREPARING (Adobe AIR 2.6 e posterior) Event.SELECT Event.DISPLAYING (propagado do menu pai)
ContextMenu	ContextMenuEvent.MENU_SELECT
ContextMenuItem	ContextMenuEvent.MENU_ITEM_SELECT Event.SELECT (AIR)

Eventos de menu Select

Adobe AIR 1.0 e posterior

Para manipular o clique de um item de menu, adicione um ouvinte de evento do evento `select` ao objeto NativeMenuItem:

```
var menuCommandX:NativeMenuItem = new NativeMenuItem("Command X");
menuCommandX.addEventListener(Event.SELECT, doCommandX)
```

Como os eventos `select` surgem em bolhas para os menus que os contêm, você também pode ouvir eventos select em um menu pai. Ao ouvir no nível de menu, é possível usar a propriedade `target` do objeto evento para determinar que comando de menu foi selecionado. O exemplo a seguir rastreia o rótulo do comando selecionado:

```
 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");
}
```

Se você estiver usando a classe ContextMenuItem, poderá ouvir o evento `select` ou o evento `menuItemSelect`. O evento `menuItemSelect` oferece informações adicionais sobre o objeto pertencente ao menu de contexto, mas não surge em bolhas para os menus que os contêm.

Exibição de eventos de menu

Adobe AIR 1.0 e posterior

Para manipular a abertura de um menu, você pode adicionar um ouvinte ao evento `displaying`, que é despachado antes de o menu ser exibido. Você pode usar o evento displaying para atualizar o menu, por exemplo, adicionando ou removendo itens ou atualizando os estados ativados ou marcados de itens individuais. Você também pode ouvir o evento `menuSelect` de um objeto ContextMenu.

No AIR 2.6 e posterior, você pode usar o evento `preparing` para atualizar um menu em resposta à exibição de um menu ou à seleção de um item com um atalho no teclado.

Exemplo de menu nativo: menu de janela e de aplicativo (AIR)

Adobe AIR 1.0 e posterior

O exemplo a seguir cria o menu mostrado em “Estrutura do menu nativo (AIR)” na página 655.

O menu foi projetado para funcionar no Windows, em que apenas os menus de janela têm suporte, e no Mac OS X, em que apenas menus de aplicativo têm suporte. Para fazer distinção, o construtor de classe MenuExample verifica as propriedades `supportsMenu` estáticas das classes NativeWindow e NativeApplication. Se `NativeWindow.supportsMenu` for `true`, o construtor cria um objeto NativeMenu para a janela e, em seguida, cria e adiciona os submenus File e Edit. Se `NativeApplication.supportsMenu` for `true`, o construtor cria e adiciona os menus File e Edit ao menu existente fornecido pelo sistema operacional Mac OS X.

O exemplo também ilustra o tratamento de evento de menu. O evento `select` é manipulado no nível de item e também no nível de menu. Cada menu na cadeia do menu que contém o item selecionado para o menu raiz responde ao evento `select`. O evento `displaying` é usado com o menu "Open Recent". Logo antes de o menu ser aberto, os itens do menu são atualizados da matriz Documentos recentes (que na realidade não é alterada neste exemplo). Embora não demonstrado nesse exemplo, você também pode ouvir eventos `displaying` em itens individuais.

```
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;
        }
    }
}
```

Capítulo 37: Ícones da barra de tarefas no AIR

Adobe AIR 1.0 e posterior

Muitos sistemas operacionais têm uma barra de tarefas, como o encaixe do Mac OS X, que pode conter um ícone representando um aplicativo. O Adobe® AIR® oferece uma interface que permite interagir com o ícone na barra de tarefas através da propriedade `NativeApplication.nativeApplication.icon`.

- Uso da bandeja do sistema e dos ícones de encaixe (Flex)
- Uso dos ícones de bandeja do sistema e do encaixe (Flash)

Mais tópicos da Ajuda

flash.desktop.NativeApplication

flash.desktop.DockIcon

flash.desktop.SystemTrayIcon

Sobre ícones na barra de tarefas

Adobe AIR 1.0 e posterior

O AIR cria o objeto `NativeApplication.nativeApplication.icon` automaticamente. O tipo de objeto é DockIcon ou SystemTrayIcon, dependendo do sistema operacional. É possível determinar qual destas subclasses InteractiveIcon é suportada pelo AIR no sistema operacional atual usando as propriedades `NativeApplication.supportsDockIcon` e `NativeApplication.supportsSystemTrayIcon`. A classe base InteractiveIcon oferece as propriedades `width`, `height` e `bitmaps`, que você pode usar para alterar a imagem utilizada para o ícone. No entanto, acessar propriedades específicas de DockIcon ou SystemTrayIcon no sistema operacional incorreto gera um erro de execução.

Para definir ou alterar a imagem usada para um ícone, crie uma matriz que contenha uma ou mais imagens e a atribua à propriedade `NativeApplication.nativeApplication.icon.bitmaps`. O tamanho dos ícones na barra de tarefas pode ser diferente nos diferentes sistemas operacionais. Para evitar a degradação da imagem devido ao dimensionamento, é possível adicionar vários tamanhos de imagem à matriz `bitmaps`. Se você fornecer mais de uma imagem, o AIR selecionará o tamanho mais próximo do tamanho de exibição atual do ícone na barra de tarefas e só o dimensionará se for necessário. O exemplo abaixo define a imagem para um ícone da barra de tarefas usando duas imagens:

```
NativeApplication.nativeApplication.icon.bitmaps =
            [bmp16x16.bitmapData, bmp128x128.bitmapData];
```

Para alterar a imagem de um ícone, atribua uma matriz que contenha a(s) nova(s) imagem(ns) à propriedade `bitmaps`. Você pode animar o ícone alterando a imagem em resposta a um evento `enterFrame` ou `timer`.

Para remover o ícone da área de notificação no Windows e no Linux ou para restaurar a aparência do ícone padrão no Mac OS X, defina `bitmaps` como uma matriz vazia:

```
NativeApplication.nativeApplication.icon.bitmaps = [];
```

Ícones de encaixe

Adobe AIR 1.0 e posterior

O AIR suporta ícones de encaixe quando `NativeApplication.supportsDockIcon` é `true`. A propriedade `NativeApplication.nativeApplication.icon` representa o ícone do aplicativo no encaixe (não um ícone de encaixe na janela).

Nota: *O AIR não permite alterar ícones de janela no encaixe do Mac OS X. Além disso, as alterações feitas no ícone de encaixe de aplicativo só são aplicadas enquanto um aplicativo está em execução — o ícone reassume a aparência normal quando o aplicativo é encerrado.*

Menus com ícone no encaixe

Adobe AIR 1.0 e posterior

É possível adicionar comandos ao menu de encaixe padrão criando um objeto NativeMenu que contenha os comandos e atribuindo o objeto à propriedade `NativeApplication.nativeApplication.icon.menu`. Os itens do menu são exibidos acima dos itens padrão do menu com ícones no encaixe.

Fazer o encaixe pular

Adobe AIR 1.0 e posterior

É possível fazer com que o ícone no encaixe pule chamando o método `NativeApplication.nativeApplication.icon.bounce()`. Se você definir o parâmetro `bounce() priority` como informativo, o ícone pulará uma vez. Se você definir o parâmetro como crítico, o ícone pulará até o usuário ativar o aplicativo. As constantes do parâmetro `priority` são definidas na classe NotificationType.

Nota: *O ícone não pula se o aplicativo já está ativo.*

Eventos de ícones no encaixe

Adobe AIR 1.0 e posterior

Quando o usuário clica em um ícone no encaixe, o objeto NativeApplication despacha um evento `invoke`. Se o aplicativo não estiver em execução, será iniciado pelo sistema. Caso contrário, o evento `invoke` será entregue à ocorrência do aplicativo em execução.

Ícones de bandeja do sistema

Adobe AIR 1.0 e posterior

O AIR oferece suporte a ícones de bandeja do sistema quando `NativeApplication.supportsSystemTrayIcon` for `true`, o que no momento só ocorre no Windows e na maioria das distribuições do Linux. No Windows e no Linux, os ícones de bandeja do sistema são exibidos na área de notificação da barra de tarefas. Nenhum ícone é exibido por padrão. Para mostrar um ícone, atribua uma matriz contendo objetos BitmapData à propriedade `bitmaps` do ícone. Para alterar a imagem do ícone, atribua uma matriz que contenha as novas imagens a `bitmaps`. Para remover o ícone, defina `bitmaps` como `null`.

Menus de ícones da bandeja do sistema

Adobe AIR 1.0 e posterior

É possível adicionar um menu ao ícone da bandeja do sistema criando-se um objeto NativeMenu e o atribuindo à propriedade `NativeApplication.nativeApplication.icon.menu` (o sistema operacional não oferece um menu padrão). Acesse o menu do ícone da bandeja do sistema clicando no ícone com o botão direito do mouse.

Dicas de ferramentas sobre ícones da bandeja do sistema

Adobe AIR 1.0 e posterior

Adicione uma dica de ferramenta a um ícone definindo a propriedade tooltip:

```
NativeApplication.nativeApplication.icon.tooltip = "Application name";
```

Eventos de ícone da bandeja do sistema

Adobe AIR 1.0 e posterior

O objeto SystemTrayIcon referenciado pela propriedade NativeApplication.nativeApplication.icon despacha um ScreenMouseEvent para eventos `click`, `mouseDown`, `mouseUp`, `rightClick`, `rightMouseDown` e `rightMouseUp`. Você pode usar esses eventos, junto com um menu de ícone, para que os usuários possam interagir com seu aplicativo quando ele não tiver janelas visíveis.

Exemplo: Criação de um aplicativo sem janelas

Adobe AIR 1.0 e posterior

O exemplo a seguir cria um aplicativo do AIR que tem um ícone na bandeja do sistema, mas nenhuma janela visível. (A propriedade `visible` do aplicativo não deve ser definida como `true` no descrito do aplicativo, ou a janela será visível quando o aplicativo for iniciado.)

```
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];
    }
  }
}
```

Nota: *Ao usar o componente WindowedApplication do Flex, defina o atributo* `visible` *da tag WindowedApplication como* `false`*. Este atributo substitui a configuração no descritor de aplicativo.*

Nota: *O exemplo pressupõe que existem arquivos de imagem chamados* `AIRApp_16.png` *e* `AIRApp_128.png` *em um subdiretório* `icons` *do aplicativo. (Arquivos de ícone de exemplo, que você pode copiar para a pasta do seu projeto, são fornecidos no SDK do AIR.)*

Ícones e botões da barra de tarefas do Windows

Adobe AIR 1.0 e posterior

As representações em forma de ícone de janelas geralmente são exibidas na área de janela de uma barra de tarefas ou de um encaixe, para que os usuários tenham fácil acesso a janelas em segundo plano ou minimizadas. O encaixe do Mac OS X exibe um ícone do seu aplicativo e um ícone de cada janela minimizada. As barras de tarefas do Microsoft Windows e do Linux exibem um botão que contém o ícone do programa e o título de cada janela de tipo normal do seu aplicativo.

Realce do botão da janela na barra de ferramentas

Adobe AIR 1.0 e posterior

Quando uma janela está em segundo plano, você pode notificar o usuário de que ocorreu um evento de interesse relacionado à janela. No Mac OS X, você pode notificar o usuário fazendo o ícone de encaixe do aplicativo pular (conforme descrito em "Fazer o encaixe pular" na página 669). No Windows e no Linux, você pode realçar o botão da barra de tarefas da janela chamando o método `notifyUser()` da ocorrência NativeWindow. O parâmetro `type` passado para o método determina a urgência da notificação:

- `NotificationType.CRITICAL`: o ícone da janela fica piscando até o usuário colocar a janela no primeiro plano.
- `NotificationType.INFORMATIONAL`: o ícone da janela fica realçado pela troca de cores.

  Nota: *No Linux, só há suporte para o tipo informativo de notificação. Passar um dos valores de tipo para a função `notifyUser()` criará o mesmo efeito.*

  A seguinte instrução realça o botão da barra de tarefas de uma janela:

  ```
  stage.nativeWindow.notifyUser(NotificationType.CRITICAL);
  ```

  Chamar o método `NativeWindow.notifyUser()` em um sistema operacional que não dá suporte a notificações de janela não produz qualquer efeito. Use a propriedade `NativeWindow.supportsNotification` para determinar se a notificação de janela é suportada.

Criação de janelas sem botões ou ícones na barra de tarefas

Adobe AIR 1.0 e posterior

No sistema operacional Windows, as janelas criadas com os tipos *utility* ou *lightweight* não aparecem na barra de tarefas. As janelas invisíveis também não.

Como a janela inicial é necessariamente do tipo *normal*, para criar um aplicativo sem nenhuma janela aparecendo na barra de tarefas, você deve fechar a janela inicial ou deixá-la invisível. Para fechar todas as janelas do seu aplicativo sem encerrá-lo, defina a propriedade `autoExit` do objeto NativeApplication como `false` antes de fechar a última janela. Para impedir que a janela inicial se torne visível, adicione `<visible>false</visible>` ao elemento `<initalWindow>` do arquivo de descrição do aplicativo (e não defina a propriedade `visible` como `true` ou chame o método `activate()` da janela).

Nas novas janelas abertas pelo aplicativo, defina a propriedade `type` do objeto NativeWindowInitOption passado para o construtor de janela como `NativeWindowType.UTILITY` ou `NativeWindowType.LIGHTWEIGHT`.

No Mac OS X, as janelas minimizadas são exibidas na barra de tarefas do encaixe. Para impedir que o ícone minimizado seja exibido, oculte a janela em vez de minimizá-la. O exemplo a seguir monitora um evento de alteração `nativeWindowDisplayState` e o cancela se a janela está sendo minimizada. O manipulador define a propriedade `visible` da janela como `false`:

```
private function preventMinimize(event:NativeWindowDisplayStateEvent):void{
    if(event.afterDisplayState == NativeWindowDisplayState.MINIMIZED){
        event.preventDefault();
        event.target.visible = false;
    }
}
```

Se uma janela está minimizada no encaixe do Mac OS X quando você define a propriedade `visible` como `false`, o ícone no encaixe não é removido. Um usuário ainda poderá clicar no ícone para fazer com que a janela reapareça.

Capítulo 38: Trabalho com o sistema de arquivos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash® Player fornece recursos básicos de leitura e gravação de arquivos, através da classe FileReference. Por razões de segurança, ao executar no Flash Player, o usuário sempre deve conceder permissão para que você possa ler ou gravar um arquivo.

O Adobe® AIR® fornece um acesso mais completo ao sistema de arquivos do computador host que está disponível no Flash Player. Usando a API do sistema de arquivos do AIR, você pode acessar e gerenciar diretórios e arquivos, criar diretórios e arquivos, gravar dados em arquivos e assim por diante.

Mais tópicos da Ajuda

flash.net.FileReference

flash.net.FileReferenceList

flash.filesystem.File

flash.filesystem.FileStream

Uso da classe FileReference

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um objeto FileReference representa um arquivo de dados em um computador cliente ou servidor. Os métodos da classe FileReference permitem que o seu aplicativo carregue e salve os arquivos de dados localmente, além de transferir dados de arquivos para ou de servidores remotos.

A classe FileReference oferece duas abordagens diferentes para carregar, transferir e salvar arquivos de dados. Desde sua introdução, a classe FileReference contém o método `browse()`, o método `upload()`, e o método `download()`. Use o método `browse()` para permitir que o usuário selecione um arquivo. Use o método `upload()` para transferir os dados do arquivo para um servidor remoto. Use o método `download()` para acessar esses dados no servidor e salvá-los em um arquivo local. A partir do Flash Player 10 e do Adobe AIR 1.5, a classe FileReference contém os métodos `load()` e `save()`. Os métodos `load()` e `save()` também permitem que você acesse e salve diretamente os arquivos locais. O uso desses métodos é semelhante ao dos métodos de nome equivalente nas classes URLLoader e Loader.

Nota: *A classe File, que estende a classe FileReference, e a classe FileStream fornecem funções adicionais para trabalhar com arquivos e com o sistema de arquivos local. As classes File e FileStream são só suportadas no AIR e não no Flash Player.*

classe FileReference

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cada objeto FileReference representa um arquivo de dados único no computador local. As propriedades da classe FileReference contêm informações sobre as seguintes características do arquivo: tamanho, tipo, nome, extensão do nome de arquivo, criador, data de criação e data de modificação.

Nota: *Há suporte para a propriedade `creator` apenas no Mac OS. Todas as outras plataformas retornam `null`.*

Nota: *A propriedade `extension` é suportada somente no Adobe AIR.*

Você pode criar uma ocorrência da classe FileReference de duas maneiras:

- Use o operador `new`, conforme apresentado no código a seguir:

```
import flash.net.FileReference;
var fileRef:FileReference = new FileReference();
```

- Chame o método `FileReferenceList.browse()`, que abre uma caixa de diálogo que solicita ao usuário selecionar um ou mais arquivos para carregar. Ele pode criar uma matriz de objetos FileReference se o usuário conseguir selecionar um ou mais arquivos.

Após criar um objeto FileReference, você pode proceder da seguinte maneira:

- Chame o método `FileReference.browse()`, que abre uma caixa de diálogo que solicita ao usuário selecionar um ou mais arquivos no sistema de arquivos local. Normalmente, isso é feito antes de uma chamada subsequente ao método `FileReference.upload()` ou `FileReference.load()`. Chame o método `FileReference.upload()` para transferir o arquivo a um servidor remoto. Chame o método `FileReference.load()` para abrir um arquivo local.
- Chame o método `FileReference.download()`. Será aberta uma caixa de diálogo pelo método `download()` para possibilitar a seleção de um local para gravação de um novo arquivo. Em seguida, os dados do servidor serão baixados e armazenados no novo arquivo.
- Chame o método `FileReference.load()`. Este método começa a carregar dados de um arquivo previamente selecionado usando o método `browse()`. O método `load()` não pode ser chamado até a operação `browse()` ser concluída (até o usuário selecionar um arquivo).
- Chame o método `FileReference.save()`. Este método abre uma caixa de diálogo e pede para o usuário escolher uma única localização de arquivo no sistema de arquivos local. Em seguida, ele salva os dados no local especificado.

Nota: *Você pode executar apenas uma ação `browse()`, `download()` ou `save()` de cada vez, porque só pode ser aberta uma caixa de diálogo a qualquer momento.*

As propriedades do objeto FileReference, como `name`, `size` ou `modificationDate` não são definidas até que ocorra um dos eventos a seguir:

- O método `FileReference.browse()` ou o método `FileReferenceList.browse()` foi chamado e o usuário selecionou um arquivo usando a caixa de diálogo.
- O método `FileReference.download()` foi chamado e o usuário especificou um novo local para os arquivos usando a caixa de diálogo.

Nota: *Ao fazer um download, apenas a propriedade `FileReference.name` é preenchida antes do download ser concluído. Após o download do arquivo, todas as propriedades estão disponíveis.*

Enquanto as chamadas dos métodos `FileReference.browse()`, `FileReferenceList.browse()`, `FileReference.download()`, `FileReference.load()` ou `FileReference.save()` estão em execução, a maioria dos players continua reproduzindo arquivos SWF, inclusive despachando eventos e executando código.

Para operações de upload e download, um arquivo SWF pode acessar arquivos apenas em seu próprio domínio, incluindo todos os domínios especificados por um arquivo de política. Você deve colocar um arquivo de política no servidor que contém o arquivo se esse servidor não está no mesmo domínio do arquivo SWF que está iniciando o upload ou o download.

See FileReference.

Carregamento de dados de arquivos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `FileReference.load()` permite carregar dados de um arquivo local na memória.

Nota: *O seu código deve chamar primeiro o método `FileReference.browse()` para que o usuário possa selecionar um arquivo a ser carregado. Essa restrição não se aplica ao conteúdo que está em execução no Adobe AIR na caixa de segurança do aplicativo*

O método `FileReference.load()` retorna imediatamente depois de ser chamado, mas os dados que estão sendo carregados não ficam disponíveis de imediato. O objeto FileReference despacha eventos para chamar métodos de ouvinte em cada etapa do processo de carregamento.

O objeto FileReference despacha os eventos a seguir durante o processo de carregamento.

- Evento `open` (`Event.OPEN`): despachado quando uma operação de carregamento é iniciada.
- Evento `progress` (`ProgressEvent.PROGRESS`): despachado periodicamente na medida em que os dados do arquivo são lidos.
- Evento `complete` (`Event.COMPLETE`): despachado quando a operação de carregamento é concluída com sucesso.
- Evento `ioError` (`IOErrorEvent.IO_ERROR`): despachado se o processo de carregamento falha devido a um erro de entrada/saída ocorrido durante a abertura ou a leitura de dados do arquivo.

Uma vez que o objeto FileReference despacha o evento complete, os dados carregados podem ser acessados como um ByteArray na propriedade `data` do objeto FileReference.

O exemplo a seguir mostra como solicitar ao usuário que selecione um arquivo e carregue os dados do arquivo na memória:

```
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.");
        }
    }
}
```

O código do exemplo primeiro cria o objeto FileReference chamado `fileRef` e depois chama o método `browse()` correspondente. Este método `browse()` abre uma caixa de diálogo que pede para o usuário selecionar um arquivo. Quando um arquivo é selecionado, o código chama o método `onFileSelected()`. Este método adiciona ouvintes dos eventos `progress` e `complete` e chama o método `load()` do objeto FileReference. Os outros métodos do manipulador citados no exemplo simplesmente geram mensagens que informam o andamento da operação de carregamento. Quando o carregamento for concluído, o aplicativo exibe o conteúdo do arquivo carregado usando o método `trace()`.

No Adobe AIR, a classe FileStream fornece uma funcionalidade extra para ler dados de um arquivo local. Consulte "Leitura e gravação de arquivos" na página 712.

Gravação de dados em arquivos locais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `FileReference.save()` permite salvar dados em um arquivo local. Ele é iniciado ao abrir uma caixa de diálogo para permitir ao usuário digitar um novo nome para o arquivo e definir um local onde o arquivo será salvo. Após a seleção do nome do arquivo e do local, os dados são gravados no novo arquivo. Quando o arquivo for salvo com êxito, as propriedades do objeto FileReference serão preenchidas com as propriedades do arquivo local.

Nota: *O código só pode chamar o método `FileReference.save()` em resposta a um evento de usuário, como um clique do mouse ou um evento de pressionamento de tecla. Caso contrário, é gerado um erro. Essa restrição não se aplica ao conteúdo que está em execução no Adobe AIR na caixa de segurança do aplicativo.*

O método `FileReference.save()` é retornado imediatamente após a chamada. Em seguida, o objeto FileReference envia eventos para chamar métodos de ouvintes em cada etapa do processo de gravação do arquivo.

O objeto FileReference despacha os eventos a seguir durante o processo de gravação do arquivo:

- Evento `select` (`Event.SELECT`): despachado quando o usuário especifica o local e o nome de arquivo para o novo arquivo a ser salvo.
- Evento `cancel` (`Event.CANCEL`): despachado quando o usuário clica no botão Cancelar da caixa de diálogo.
- Evento `open` (`Event.OPEN`): despachado quando uma operação de salvamento é iniciada.
- Evento `progress` (`ProgressEvent.PROGRESS`): despachado periodicamente na medida em que os dados são gravados no arquivo.
- Evento `complete` (`Event.COMPLETE`): despachado quando a operação de gravação é concluída com sucesso.
- Evento `ioError` (`IOErrorEvent.IO_ERROR`): despachado se o processo de salvamento falha devido a um erro de entrada/saída ocorrido durante a tentativa de salvar dados no arquivo.

O tipo de objeto transmitido no parâmetro `data` do método `FileReference.save()` determina como os dados são gravados no arquivo:

- Se for um valor String, os dados serão salvos como um arquivo de texto na codificação UTF-8.
- Se for um objeto XML, eles serão gravados em um arquivo no formato XML com toda a formatação preservada.
- Se for um objeto ByteArray, seu conteúdo será gravado diretamente no arquivo, sem conversão.
- Se for outro objeto, o método `FileReference.save()` chamará o método `toString()` do objeto e salvará o valor String resultante em um arquivo de texto UTF-8. Se não for possível chamar o método `toString()` do objeto, será gerado um erro.

Se o valor do parâmetro `data` for `null`, será gerado um erro.

O código a seguir amplia o exemplo anterior do método `FileReference.load()`. Após a leitura dos dados do arquivo, esse exemplo solicita ao usuário um nome de arquivo e, em seguida, salva os dados em um novo arquivo:

```
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.");
        }
    }
}
```

Quando todos os dados forem carregados do arquivo, o código chamará o método `onComplete()`. O método `onComplete()` remove os ouvintes dos eventos de carregamento e, em seguida, chama o método `saveFile()`. O método `saveFile()` chama o método `FileReference.save()`. O método `FileReference.save()` abre uma nova caixa de diálogo na qual o usuário pode digitar um novo nome e um local para salvar o arquivo. Os métodos restantes do ouvinte de eventos rastreiam o andamento do processo de gravação do arquivo até que seja concluído.

No Adobe AIR, a classe FileStream fornece uma funcionalidade extra para gravar dados em um arquivo local. Consulte "Leitura e gravação de arquivos" na página 712.

Upload de arquivos em um servidor

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para carregar arquivos para um servidor, chame primeiro o método `browse()` para permitir que um usuário selecione um ou mais arquivos. Em seguida, quando o método `FileReference.upload()` é chamado, o arquivo selecionado é transferido para o servidor. Se o usuário seleciona vários arquivos usando o método `FileReferenceList.browse()`, o Flash Player cria uma matriz dos arquivos selecionados chamada `FileReferenceList.fileList`. Em seguida, você pode usar o método `FileReference.upload()` para carregar cada arquivo individualmente.

Nota: *O uso do método `FileReference.browse()` permite carregar apenas arquivos únicos. Para permitir que um usuário carregue vários arquivos, use o método `FileReferenceList.browse()`.*

Por padrão, a caixa de diálogo de seletor de arquivos do sistema permite que os usuários selecionem qualquer tipo de arquivo do computador local. Os desenvolvedores podem especificar um ou mais filtros do tipo de arquivo personalizado usando a classe FileFilter e transmitindo uma matriz de instâncias de filtros de arquivo para o método `browse()`:

```
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);
```

Quando o usuário seleciona os arquivos e clica no botão Abrir no seletor de arquivos do sistema, o evento `Event.SELECT` é enviado. Se o método `FileReference.browse()` for usado para selecionar um arquivo a ser carregado, o seguinte código enviará o arquivo a um servidor Web:

```
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");
}
```

Você pode enviar dados ao servidor com o método `FileReference.upload()` usando as propriedades `URLRequest.method` e `URLRequest.data` para enviar variáveis que usam os métodos `POST` ou `GET`.

Quando você tenta carregar um arquivo usando o método `FileReference.upload()`, os seguintes eventos são despachados:

- Evento `open` (`Event.OPEN`): despachado quando uma operação de upload é iniciada.
- Evento `progress` (`ProgressEvent.PROGRESS`): despachado periodicamente na medida em que os dados do arquivo são carregados.
- Evento `complete` (`Event.COMPLETE`): despachado quando a operação de upload é concluída com sucesso.
- Evento `httpStatus` (`HTTPStatusEvent.HTTP_STATUS`): despachado quando há uma falha no processo de upload devido a um erro HTTP.

- Evento `httpResponseStatus` (`HTTPStatusEvent.HTTP_RESPONSE_STATUS`): Despachado se uma chamada do método `upload()` ou `uploadUnencoded()` tentar acessar os dados através de HTTP e o Adobe AIR puder detectar e retornar o código de status da solicitação.
- Evento `securityError` (`SecurityErrorEvent.SECURITY_ERROR`): despachado quando há uma falha na operação de upload devido a uma violação de segurança.
- Evento `uploadCompleteData` (`DataEvent.UPLOAD_COMPLETE_DATA`): despachado quando os dados são recebidos do servidor após um upload bem-sucedido.
- Evento `ioError` (`IOErrorEvent.IO_ERROR`): despachado se houver falha no processo de upload por qualquer um dos seguintes motivos:
  - Ocorreu um erro de entrada/saída enquanto o Flash Player estava lendo, gravando ou transmitindo o arquivo.
  - O SWF tentou carregar um arquivo para um servidor que requer autenticação (como um nome de usuário e senha). Durante o upload, o Flash Player não fornece um meio para os usuários inserirem senhas.
  - O parâmetro `url` contém um protocolo inválido. O método `FileReference.upload()` deve usar HTTP ou HTTPS.

O Flash Player não oferece suporte completo para servidores que requerem autenticação. Apenas arquivos SWF que estão em execução em um navegador que usa o plug-in do navegador ou o controle Microsoft ActiveX® podem fornecer uma caixa de diálogo para solicitar que o usuário digite um nome de usuário e senha para autenticação e apenas para downloads. Ocorre falha na transferência de arquivos, para uploads que usam o plug-in ou o controle ActiveX, ou upload/download que usa player autônomo ou externo.

Para criar um script de servidor no ColdFusion para aceitar um upload de arquivo do Flash Player, você pode usar um código semelhante a este:

```
<cffile action="upload" filefield="Filedata" destination="#ExpandPath('./')#"
nameconflict="OVERWRITE" />
```

Esse código ColdFusion carrega o arquivo enviado pelo Flash Player e salva-o no mesmo diretório que o modelo ColdFusion, substituindo qualquer arquivo com o mesmo nome. O código anterior mostra a quantidade mínima básica de código necessária para aceitar um upload de arquivo. Esse script não deve ser usado em um ambiente de produção. Em condições ideais, adicione validação de dados para garantir que os usuários carreguem apenas tipos de arquivo aceitos, como uma imagem, em vez de um script do lado do servidor potencialmente perigoso.

O código a seguir demonstra uploads de arquivo usando PHP e inclui validação de dados. O script limita o número de arquivos carregados no diretório de upload até 10, garante que o arquivo tenha menos do que 200 KB e permite que apenas arquivos JPEG, GIF ou PNG sejam carregados e salvos no sistema de arquivos.

```
<?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;
    }
}
?>
```

Você pode transmitir variáveis adicionais para o script de upload usando o método de solicitação `POST` ou o`GET`. Para enviar variáveis `POST` adicionais ao script de upload, use o seguinte código:

```
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");
}
```

O exemplo anterior cria um objeto URLVariables que você transmite para o script do lado do servidor remoto. Em versões anteriores do ActionScript, era possível transmitir variáveis para o script de upload de servidor transmitindo valores na string da consulta. O ActionScript 3.0 permite transmitir variáveis para o script remoto usando um objeto URLRequest, o que permite que você transmita dados usando o método `POST` ou o `GET`. Isso, por sua vez, faz com que a transmissão de conjuntos de dados maiores seja mais fácil e mais limpa. Para especificar se a variáveis são transmitidas usando o método de solicitação `GET` ou `POST`, defina a propriedade `URLRequest.method` como `URLRequestMethod.GET` ou `URLRequestMethod.POST`, respectivamente.

O ActionScript 3.0 também permite substituir o nome do campo do arquivo de upload `Filedata` padrão fornecendo um segundo parâmetro para o método `upload()`, conforme demonstrado no exemplo anterior (que substituiu o valor padrão `Filedata` por `Custom1`).

Por padrão, o Flash Player não tenta enviar um upload de teste, embora seja possível substituir esse padrão transmitindo um valor `true` como o terceiro parâmetro para o método `upload()`. A finalidade do upload de teste é verificar se o upload do arquivo real será bem-sucedido e se a autenticação do servidor, se necessária, terá êxito.

Nota: *No momento, um upload de teste ocorre apenas em Flash Players com base no Windows.*

O script de servidor que manipula o upload do arquivo deve esperar uma solicitação `POST` HTTP com os seguintes elementos:

- `Content-Type` com um valor de `multipart/form-data`.
- `Content-Disposition` com um atributo `name` definido como "`Filedata`" e um atributo `filename` definido como o nome do arquivo original. É possível especificar um atributo `name` personalizado, transmitindo um valor para o parâmetro `uploadDataFieldName` no método `FileReference.upload()`.
- O conteúdo binário do arquivo.

Esta é uma solicitação `POST` HTTP de amostra:

```
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,,,)
```

A seguinte solicitação `POST` HTTP de amostra envia três variáveis `POST`: `api_sig`, `api_key` e `auth_token` e usa um valor de nome de campo de dados de upload personalizado de `"photo"`:

```
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"

XXXXXXXXXXXXXXXXXXXXXXX
------------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--
```

Download de arquivos de um servidor

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível permitir que os usuários baixem arquivos de um servidor usando o método `FileReference.download()` que utiliza dois parâmetros: `request` e `defaultFileName`. O primeiro parâmetro é o objeto URLRequest que contém a URL do arquivo a ser baixado. O segundo parâmetro é opcional, ele permite especificar um nome de arquivo padrão que aparece na caixa de diálogo do arquivo de download. Se você omitir o segundo parâmetro, `defaultFileName`, o nome de arquivo da URL especificada será usado.

O código a seguir baixa um arquivo denominado index.xml do mesmo diretório que o arquivo SWF:

```
var request:URLRequest = new URLRequest("index.xml");
var fileRef:FileReference = new FileReference();
fileRef.download(request);
```

Para definir o nome padrão como currentnews.xml, em vez de index.xml, especifique o parâmetro `defaultFileName`, conforme mostrado no seguinte snippet:

```
var request:URLRequest = new URLRequest("index.xml");
var fileToDownload:FileReference = new FileReference();
fileToDownload.download(request, "currentnews.xml");
```

Renomear um arquivo poderá ser útil se o nome de arquivo do servidor não for intuitivo ou gerado pelo servidor. Também é bom especificar explicitamente o parâmetro `defaultFileName` ao baixar um arquivo usando um script do lado do servidor, em vez de baixar o arquivo diretamente. Por exemplo, você precisa especificar o parâmetro `defaultFileName`, se tiver um script do lado do servidor que baixe arquivos específicos com base em variáveis de URL transmitidas para ele. Caso contrário, o nome padrão do arquivo baixado será o nome do script do lado do servidor.

Os dados podem ser enviados ao servidor usando o método `download()` anexando parâmetros à URL para análise pelo script do servidor. O seguinte snippet do ActionScript 3.0 baixa um documento com base em quais parâmetros são transmitidos para um script ColdFusion:

```
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.");
            }
        }
    }
}
```

O código a seguir demonstra o script ColdFusion, download.cfm, que baixa um de dois arquivos do servidor, dependendo do valor de uma variável de URL:

```
<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>
```

classe FileReferenceList

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe FileReferenceList permite que o usuário selecione um ou mais arquivos a serem carregados para um script do lado do servidor. O upload de arquivo é manipulado pelo método `FileReference.upload()` que deve ser chamado em cada arquivo selecionado pelo usuário.

O código a seguir cria dois objetos FileFilter (`imageFilter` e `textFilter`) e transmite-os em uma matriz para o método `FileReferenceList.browse()`. Isso faz com que a caixa de diálogo de arquivo do sistema operacional exiba dois filtros possíveis para tipos de arquivos.

```
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.");
}
```

Permitir que o usuário selecione e carregue um ou mais arquivos usando a classe FileReferenceList é o mesmo que usar `FileReference.browse()` para selecionar arquivos, embora o FileReferenceList permita selecionar mais de um arquivo. O upload de vários arquivos requer que você carregue cada um dos arquivos selecionados usando `FileReference.upload()`, conforme mostrado no código a seguir:

```
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");
}
```

Como o evento `Event.COMPLETE` é adicionado a cada objeto FileReference individual na matriz, o Flash Player chama o método `completeHandler()` quando o upload de cada arquivo individual é concluído.

Usar a API do sistema de arquivos do AIR

Adobe AIR 1.0 e posterior

A API do sistema de arquivos Adobe AIR inclui as seguintes classes:

- Arquivo
- FileMode
- FileStream

A API do sistema de arquivos permite fazer o seguinte (e muito mais):

- Copiar, criar, excluir e mover arquivos e diretórios
- Obtenção de informações sobre arquivos e diretórios
- Ler e gravar arquivos

Noções básicas do arquivo AIR

Adobe AIR 1.0 e posterior

Para ver uma explicação rápida e exemplos de código para trabalhar com o sistema de arquivos no AIR, consulte os seguintes artigos de apresentação rápida do Adobe Developer Connection:

- Criando um editor de arquivo de texto (Flash)
- Criação de editor de arquivo de texto (Flex)
- Criação de aplicativo de pesquisa em diretório (Flex)
- Leitura e gravação de um arquivo de preferências XML (Flex)
- Compactação de arquivos e dados (Flex)

O Adobe AIR oferece classes que você pode usar para acessar, criar e gerenciar arquivos e pastas. Essas classes, contidas no pacote flash.filesystem, são usadas da seguinte forma:

Classes File	Descrição
File	O objeto File representa um caminho para um arquivo ou um diretório. Você usa o objeto File para criar um indicador para um arquivo ou pasta, iniciando a interação com o arquivo ou a pasta.
FileMode	A classe FileMode define constantes de string usadas no parâmetro `fileMode` dos métodos `open()` e `openAsync()` da classe FileStream. O parâmetro `fileMode` desses métodos determina os recursos disponíveis para o objeto FileStream depois que o arquivo é aberto, o que inclui gravação, leitura, acréscimo e atualização.
FileStream	O objeto FileStream é usado para abrir arquivos de leitura e gravação. Depois que você criar o objeto File apontando para um arquivo novo ou para um já existente, passe esse ponteiro para o objeto FileStream, de modo a poder abri-lo e, em seguida, manipular dados no arquivo.

Alguns métodos na classe File têm versões síncronas e assíncronas:

- `File.copyTo()` e `File.copyToAsync()`
- `File.deleteDirectory()` e `File.deleteDirectoryAsync()`
- `File.deleteFile()` e `File.deleteFileAsync()`
- `File.getDirectoryListing()` e `File.getDirectoryListingAsync()`
- `File.moveTo()` e `File.moveToAsync()`
- `File.moveToTrash()` e `File.moveToTrashAsync()`

Além disso, as operações FileStream funcionam síncrona ou assincronamente, dependendo de como o objeto FileStream abre o arquivo: chamando o método `open()` ou o método `openAsync()`.

As versões assíncronas permitem iniciar processos que são executados em segundo plano e despacham eventos quando concluídos (ou quando ocorrerem eventos de erro). Outro código pode ser executado enquanto esses processos de segundo plano assíncronos estão ocorrendo. Nas versões assíncronas das operações, você deve configurar funções do ouvinte de eventos, usando o método `addEventListener()` do objeto File ou FileStream que chama a função.

As versões síncronas permitem escrever códigos mais simples que não se baseiam na configuração de ouvintes de evento. No entanto, como nenhum outro código pode ser executado enquanto o método sincrônico está sendo executado, processos importantes, como renderização e animação de objeto de exibição, podem ser retardados.

Trabalho com objetos File no AIR

Adobe AIR 1.0 e posterior

O objeto File é um ponteiro para um arquivo ou diretório no sistema de arquivos.

A classe File estende a classe FileReference. A classe FileReference, disponível no Adobe® Flash® Player, bem como no AIR, representa um ponteiro para um arquivo, mas a classe File adiciona propriedades e métodos que não são expostos no Flash Player (em um SWF em execução no navegador), devido a considerações sobre segurança.

Sobre a classe File

Adobe AIR 1.0 e posterior

Você pode usar a classe File para o seguinte:

- Obter o caminho para diretórios especiais, incluindo o diretório do usuário, o diretório de documentos do usuário, o diretório do qual o aplicativo foi iniciado e o diretório de aplicativo.
- Cópia de arquivos e diretórios
- Movimentação de arquivos e diretórios.
- Exclusão de arquivos e diretórios (ou movimentação para a lixeira)
- Lista de arquivos e diretórios contidos em um diretório
- Criação de arquivos e pastas temporários

Depois que o objeto File aponta para um caminho de arquivo, você pode usá-lo para ler e gravar dados de arquivo, usando a classe FileStream.

O objeto File pode apontar para o caminho de um arquivo ou diretório que ainda não existe. Você pode usar esse objeto File na criação de arquivos ou diretórios.

Caminhos de objetos File

Adobe AIR 1.0 e posterior

Cada objeto File tem duas propriedades que definem cada uma o caminho do objeto:

Propriedade	Descrição
`nativePath`	Especifica o caminho específico de plataforma para um arquivo. Por exemplo, no Windows o caminho pode ser "c:\Sample directory\test.txt", enquanto no Mac OS pode ser "/Sample directory/test.txt". A propriedade `nativePath` usa o caractere de barra invertida (\) como o caractere separador de diretório no Windows e o caractere de barra (/) no Mac OS e no Linux.
`url`	Ela pode usar o esquema URL de arquivo para apontar para um arquivo. Por exemplo, no Windows o caminho pode ser "file:///c:/Sample%20directory/teste.txt", enquanto no Mac OS pode ser "file:///Sample%20directory/teste.txt". O tempo de execução inclui outros esquemas especiais de URL além de `file` e são descritos em "Esquemas URL do AIR com suporte" na página 699

A classe File inclui propriedades estáticas de indicação de diretórios padrão no Mac OS, no Windows e no Linux. Essas propriedades incluem:

- `File.applicationStorageDirectory` — um diretório de armazenamento exclusivo para cada aplicativo do AIR instalado. Este diretório é um lugar apropriado para armazenar ativos dinâmicos do aplicativo e preferências do usuário. Considere o armazenamento de grandes volumes de dados em outro local.

  No Android, o diretório de armazenamento do aplicativo é removido quando o aplicativo é desinstalado ou o usuário opta por limpar os dados do aplicativo, mas isso não ocorre em outras plataformas.

- `File.applicationDirectory` – o diretório em que o aplicativo está instalado (junto com quaisquer ativos instalados). Em alguns sistemas operacionais, o aplicativo é armazenado em um único arquivo de pacote e não em um diretório físico. Nesse caso, o conteúdo pode não ficar acessível por meio do caminho nativo. O diretório do aplicativo é somente leitura.
- `File.desktopDirectory` – o diretório de área de trabalho do usuário. Se a plataforma não definir um diretório de área de trabalho, será usado outro local no sistema de arquivos.
- `File.documentsDirectory` – o diretório de documentos do usuário. Se a plataforma não definir um diretório de documentos, será usado outro local no sistema de arquivo.
- `File.userDirectory` – o diretório do usuário. Se a plataforma não definir um diretório do usuário, será usado outro local no sistema de arquivo.

Nota: *Quando a plataforma não definir locais padrão para os diretórios de área de trabalho, de documentos ou do usuário,* `File.documentsDirectory`*,* `File.desktopDirectory` *e* `File.userDirectory` *podem fazer referência ao mesmo diretório.*

Estas propriedades têm valores diferentes em diferentes sistemas operacionais. Por exemplo, o Mac e o Windows têm caminhos nativos diferentes para o diretório de área de trabalho do usuário. Entretanto, a propriedade `File.desktopDirectory` aponta para um caminho de diretório apropriado em cada plataforma. Para gravar aplicativos que funcionam bem em várias plataformas, use essas propriedades como base para fazer referência a outros diretórios e arquivos usados pelo aplicativo. Use o método `resolvePath()` para refinar o caminho. Por exemplo, esse código aponta para o arquivo preferences.xml no diretório de armazenamento do aplicativo:

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("preferences.xml");
```

Embora a classe File permita apontar para um caminho específico de arquivo, fazer isso pode levar a aplicativos que não funcionam em várias plataformas. Por exemplo, o caminho C:\Documents and Settings\joe\ só funciona no Windows. Por essas razões, é melhor usar as propriedades estáticas da classe File, como `File.documentsDirectory`.

Locais de diretórios comuns

Plataforma	Tipo de diretório	Local típico do sistema de arquivos
Android	Aplicativo	`/data/data/`
	Armazenamento do aplicativo	`/data/data/air.applicationID/filename/Local Store`
	Cache	/data/data/applicationID/cache
	Área de trabalho	`/mnt/sdcard`
	Documentos	`/mnt/sdcard`
	Temporário	`/data/data/applicationID/cache/FlashTmp.randomString`
	Usuário	`/mnt/sdcard`
iOS	Aplicativo	`/var/mobile/Applications/uid/filename.app`
	Armazenamento do aplicativo	`/var/mobile/Applications/uid/Library/Application Support/applicationID/Local Store`
	Cache	/var/mobile/Applications/uid/Library/Caches
	Área de trabalho	não acessível
	Documentos	`/var/mobile/Applications/uid/Documents`
	Temporário	`/private/var/mobile/Applications/uid/tmp/FlashTmpNNN`
	Usuário	não acessível
Linux	Aplicativo	`/opt/filename/share`
	Armazenamento do aplicativo	`/home/userName/.appdata/applicationID/Local Store`
	Área de trabalho	`/home/userName/Desktop`
	Documentos	`/home/userName/Documents`
	Temporário	`/tmp/FlashTmp.randomString`
	Usuário	`/home/userName`
Mac	Aplicativo	`/Applications/filename.app/Contents/Resources`
	Armazenamento do aplicativo	`/Users/`***`userName`***`/Library/Preferences/`***`applicationid`***`/Local Store` (AIR 3.2 e anterior) ***`path`***`/Library/Application Support/`***`applicationid`***`/Local Store` (AIR 3.3 e posterior), onde o caminho é `/Users/`***`userName`***`/Library/Containers/`***`bundle-id`***`/Data` (ambiente seguro) ou `/Users/userName` (ao executar fora de um ambiente seguro)
	Cache	/Users/userName/Library/Caches
	Área de trabalho	`/Users/`***`userName`***`/Desktop`
	Documentos	`/Users/`***`userName`***`/Documents`
	Temporário	`/private/var/folders/JY/randomString/TemporaryItems/FlashTmp`
	Usuário	`/Users/`***`userName`***

Plataforma	Tipo de diretório	Local típico do sistema de arquivos
Windows	Aplicativo	`C:\Arquivos de programa\filename`
	Armazenamento do aplicativo	`C:\Documents and settings\userName\ApplicationData\applicationID\Local Store`
	Cache	C:\Documents and settings\userName\Configurações locais\Temp
	Área de trabalho	`C:\Documents and settings\userName\Área de trabalho`
	Documentos	`C:\Documents and settings\userName\Meus documentos`
	Temporário	`C:\Documents and settings\userName\Configurações locais\Temp\randomString.tmp`
	Usuário	`C:\Documents and settings\userName`

Os caminhos nativos reais desses diretórios variam com base no sistema operacional e na configuração do computador. Os caminhos mostrados nesta tabela são exemplos típicos. Você deve sempre usar as propriedades estáticas apropriadas da classe File para fazer referência a esses diretórios, de modo que o seu aplicativo funcione corretamente em qualquer plataforma. No aplicativo do AIR real, os valores de `applicationID` e `filename` mostrados na tabela são obtidos do descritor do aplicativo. Se você especificar uma ID de editor no descritor do aplicativo, a ID de editor será anexada à ID do aplicativo nesses caminhos. O valor de `userName` é o nome da conta do usuário que faz a instalação.

Como apontar um objeto File para um diretório

Adobe AIR 1.0 e posterior

Há maneiras diferentes de configurar o objeto File para apontar para um diretório.

Apontar para o diretório inicial do usuário

Adobe AIR 1.0 e posterior

Você pode apontar o objeto File para o diretório inicial do usuário. O código a seguir configura o objeto File para apontar para o subdiretório AIR Test do diretório inicial:

```
var file:File = File.userDirectory.resolvePath("AIR Test");
```

Apontar para o diretório documentos do usuário

Adobe AIR 1.0 e posterior

Você pode apontar o objeto File para o diretório documentos do usuário. O código a seguir configura o objeto File para apontar para o subdiretório AIR Test do diretório documentos:

```
var file:File = File.documentsDirectory.resolvePath("AIR Test");
```

Apontar para o diretório da área de trabalho

Adobe AIR 1.0 e posterior

Você pode apontar o objeto File para a área de trabalho. O código a seguir configura o objeto File para apontar para o subdiretório AIR Test da área de trabalho:

```
var file:File = File.desktopDirectory.resolvePath("AIR Test");
```

Apontar para o diretório de armazenamento do aplicativo

Adobe AIR 1.0 e posterior

Você pode apontar o objeto File para o diretório de armazenamento do aplicativo. Para cada aplicativo do AIR, há um caminho associado exclusivo que define o diretório de armazenamento do aplicativo. Esse diretório é exclusivo de cada aplicativo e usuário. Você pode usar esse diretório para armazenar dados específicos do usuário e do aplicativo (como dados do usuário ou arquivos de preferências). Por exemplo, o código a seguir aponta o objeto File para um arquivo de preferências, prefs.xml, contido no diretório de armazenamento do aplicativo:

```
var file:File = File.applicationStorageDirectory;
file = file.resolvePath("prefs.xml");
```

Normalmente, o local do diretório de armazenamento do aplicativo tem base no nome do usuário e no ID do aplicativo. Os seguintes locais no sistema de arquivos são fornecidos aqui para ajudar a depurar o aplicativo. Você deve sempre usar a propriedade `File.applicationStorage` ou o esquema de URI `app-storage:` para resolver arquivos nesse diretório:

- No Mac OS — varia com a versão do AIR:

  AIR 3.2 e anterior: `/Users/nome do usuário/Library/Preferences/`*`applicationID`*`/Local Store/`

  AIR 3.3 e posterior: *`path`*`/Library/Application Support/`*`applicationID`*`/Local Store`, onde *`caminho`* é `/Users/`*`username`*`/Library/Containers/`*`bundle-id`*`/Data` (ambiente seguro) ou `/Users/`*`username`* (ao executar fora de um ambiente seguro)

  Por exemplo (AIR 3.2):

  ```
  /Users/babbage/Library/Preferences/com.example.TestApp/Local Store
  ```

- No Windows: no diretório Documents and Settings, em:

  C:\Documents and Settings\user name`\Application Data\`*applicationID*`\Local Store\`

  Por exemplo:

  ```
  C:\Documents and Settings\babbage\Application Data\com.example.TestApp\Local Store
  ```

- No Linux - In:

  `/home/`*user name*`/.appdata/`*applicationID*`/Local Store/`

  Por exemplo:

  ```
  /home/babbage/.appdata/com.example.TestApp/Local Store
  ```

- No Android – Em:

  /data/data/`androidPackageID`/`applicationID`/`Local Store`

  Por exemplo:

  ```
  /data/data/air.com.example.TestApp/com.example.TestApp/Local Store
  ```

Nota: *Se um aplicativo possuir um ID de publicação, o ID de publicação também é utilizado como parte do caminho do diretório de armazenamento do aplicativo.*

A URL (e a propriedade `url`) de um objeto File criado com `File.applicationStorageDirectory` usa o esquema de URL `app-storage` (consulte "Esquemas URL do AIR com suporte" na página 699), como a seguir:

```
var dir:File = File.applicationStorageDirectory;
dir = dir.resolvePath("preferences");
trace(dir.url); // app-storage:/preferences
```

Apontar para o diretório do aplicativo

Adobe AIR 1.0 e posterior

Você pode apontar o objeto File para o diretório em que o aplicativo foi instalado, conhecido como o diretório do aplicativo. Você pode fazer referência a esse diretório, usando a propriedade `File.applicationDirectory`. Você pode usar esse diretório para examinar o arquivo do descritor do aplicativo ou outros recursos instalados com o aplicativo. Por exemplo, o código a seguir aponta o objeto File para um diretório chamado *images* no diretório do aplicativo:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
```

A URL (e propriedade `url`) de um objeto File criado com `File.applicationDirectory` usa o esquema de URL `app` (consulte "Esquemas URL do AIR com suporte" na página 699), como a seguir:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
trace(dir.url); // app:/images
```

Nota: *No Android, os arquivos no pacote de aplicativos não ficam acessíveis pelo `nativePath`. A propriedade `nativePath` é uma string vazia. Use sempre o URL para acessar arquivos no diretório do aplicativo em vez de usar um caminho nativo.*

Apontar para o diretório do cache

Adobe AIR 3.6 e posterior

Você pode apontar um objeto File para o diretório de cache ou temporário do sistema operacional utilizando a propriedade `File.cacheDirectory`. Trata-se de um diretório que contém arquivos temporários que não são necessários para que o aplicativo funcione e não causarão problemas ou perda de dados para o usuário caso sejam excluídos.

Na maioria dos sistemas operacionais, o diretório de cache é um diretório temporário. No iOS, o diretório de cache corresponde ao diretório de caches da biblioteca de aplicativos. Os arquivos neste diretório não são copiados para o armazenamento online, podendo ser excluídos pelo sistema operacional se o espaço de armazenamento disponível do dispositivo não for suficiente. Para obter mais informações, consulte "Controle do backup de arquivos e do armazenamento em cache" na página 699.

Apontando para a raiz do sistema de arquivos

Adobe AIR 1.0 e posterior

O método `File.getRootDirectories()` lista todos os volumes de raiz, como C: e volumes montados em um computador Windows. No Mac OS e no Linux, esse método sempre retorna o diretório raiz exclusivo do computador (o diretório "/"). O método `StorageVolumeInfo.getStorageVolumes()` fornece informações mais detalhadas sobre os volumes montados de armazenamento (consulte "Trabalho com volumes de armazenamento" na página 710).

Nota: *A raiz do sistema de arquivos não é legível no Android. Um objeto File que faz referência ao diretório com o caminho nativo "/" é retornado, mas as propriedades do objeto não têm valores precisos. Por exemplo, `spaceAvailable` é sempre 0.*

Como apontar para um diretório explícito

Adobe AIR 1.0 e posterior

Você pode apontar o objeto File para um diretório explícito, definindo a propriedade `nativePath` do objeto File, conforme o seguinte exemplo (no Windows):

```
var file:File = new File();
file.nativePath = "C:\\AIR Test";
```

Importante: apontar para um caminho explícito dessa forma pode levar a um código que não funciona em várias plataformas. Por exemplo, o exemplo anterior só funciona no Windows. Você pode usar as propriedades estáticas do objeto File, como `File.applicationStorageDirectory`, para localizar um diretório que funciona em várias plataformas. Use o método `resolvePath()` (consulte a próxima seção) para ir até um caminho relativo.

Navegação para caminhos relativos

Adobe AIR 1.0 e posterior

Você pode usar o método `resolvePath()` para obter um caminho relativo a outro caminho determinado. Por exemplo, o código a seguir configura o objeto File para apontar para o subdiretório "AIR Test" do diretório inicial do usuário:

```
var file:File = File.userDirectory;
file = file.resolvePath("AIR Test");
```

Você também pode usar a propriedade `url` do objeto File para apontar para um diretório com base em uma sequência de URL, conforme segue:

```
var urlStr:String = "file:///C:/AIR Test/";
var file:File = new File()
file.url = urlStr;
```

Para obter mais informações, consulte “Modificação de caminhos do File” na página 699.

Permissão para que o usuário navegue para selecionar um diretório

Adobe AIR 1.0 e posterior

A classe File inclui o método `browseForDirectory()`, que apresenta uma caixa de diálogo de sistema na qual o usuário pode selecionar um diretório para atribuir ao objeto. O método `browseForDirectory()` é assíncrono. O objeto File despachará um evento `select` se o usuário selecionar um diretório e clicar no botão Abrir ou despachará um evento `cancel` se o usuário clicar no botão Cancelar.

Por exemplo, o código a seguir permite que o usuário selecione um diretório e fornece o caminho do diretório na seleção:

```
var file:File = new File();
file.addEventListener(Event.SELECT, dirSelected);
file.browseForDirectory("Select a directory");
function dirSelected(e:Event):void {
    trace(file.nativePath);
}
```

Nota: *No Android, não há suporte para o método `browseForDirectory()`. Chamar esse método não tem nenhum efeito. Um evento cancel é despachado imediatamente. Em vez disso, para permitir que os usuários selecionem um diretório, é preciso usar uma caixa de diálogo personalizada definida pelo aplicativo.*

Como apontar para o diretório do qual o aplicativo foi chamado

Adobe AIR 1.0 e posterior

Você pode obter a localização do diretório do qual o aplicativo foi chamado, marcando a propriedade `currentDirectory` do objeto InvokeEvent despachada quando o aplicativo é chamado. Para obter detalhes, consulte"Captura de argumentos de linha de comando" na página 908.

Como apontar um objeto File para um arquivo

Adobe AIR 1.0 e posterior

Há maneiras diferentes de definir o arquivo para o qual o objeto File aponta.

Apontar para um caminho de arquivo explícito

Adobe AIR 1.0 e posterior

Importante: apontar para um caminho explícito pode levar a um código que não funciona em várias plataformas. Por exemplo, o caminho C:/foo.txt só funciona no Windows. Você pode usar as propriedades estáticas do objeto File, como `File.applicationStorageDirectory`, para localizar um diretório que funciona em várias plataformas. Use o método `resolvePath()` (consulte "Modificação de caminhos do File" na página 699 para ir até um caminho relativo.

Você pode usar a propriedade `url` do objeto File para apontá-lo para um arquivo ou diretório com base em uma sequência de URL, conforme segue:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File()
file.url = urlStr;
```

Você também pode passar a URL para a função de construtor `File()`, como no seguinte:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File(urlStr);
```

A propriedade `url` sempre retorna a versão codificada de URI da URL (por exemplo, espaços em branco são substituídos por `"%20`):

```
file.url = "file:///c:/AIR Test";
trace(file.url); // file:///c:/AIR%20Test
```

Você também pode usar a propriedade `nativePath` do objeto File para definir um caminho explícito. Por exemplo, o código a seguir, quando executado em um computador com o Windows, define um objeto File para o arquivo test.txt no subdiretório AIR Test da unidade C:

```
var file:File = new File();
file.nativePath = "C:/AIR Test/test.txt";
```

Você também pode passar esse caminho para a função de construtor `File()`, como no seguinte:

```
var file:File = new File("C:/AIR Test/test.txt");
```

Use o caractere barra (/) como o delimitador de caminho para a propriedade `nativePath`. No Windows, você também pode usar o caractere de barra invertida (\), mas fazer isso leva a aplicativos que não funcionam em várias plataformas.

Para obter mais informações, consulte "Modificação de caminhos do File" na página 699.

Enumeração de arquivos em um diretório

Adobe AIR 1.0 e posterior

Você pode usar o método `getDirectoryListing()` do objeto File para obter uma matriz de objetos File apontando para arquivos e subdiretórios no nível raiz de um diretório. Para obter mais informações, consulte "Enumeração de diretórios" na página 706.

Permissão para que o usuário navegue para selecionar um arquivo

Adobe AIR 1.0 e posterior

A classe File inclui os seguintes métodos que apresentam uma caixa de diálogo de sistema na qual o usuário pode selecionar um arquivo para atribuir ao objeto:

- `browseForOpen()`
- `browseForSave()`
- `browseForOpenMultiple()`

Cada um desses métodos é assíncrono. Os métodos `browseForOpen()` e `browseForSave()` despacham o evento select quando o usuário seleciona um arquivo (ou caminho de destino, no caso de browseForSave()). Com os métodos `browseForOpen()` e `browseForSave()`, na seleção, o objeto File de destino aponta para os arquivos selecionados. O método `browseForOpenMultiple()` despacha um evento `selectMultiple` quando o usuário seleciona arquivos. O evento `selectMultiple` é do tipo FileListEvent, que tem a propriedade `files`, que é uma matriz de objetos File (apontando para os arquivos selecionados).

Por exemplo, o código a seguir apresenta para o usuário a caixa de diálogo "Abrir", na qual é possível selecionar um arquivo:

```
 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);
}
```

Se o aplicativo tiver outra caixa de diálogo de navegação aberta quando você chamar o método de navegação, o tempo de execução emitirá uma exceção Erro.

Nota: *No Android, somente arquivos de imagem, vídeo e áudio podem ser selecionados com os métodos `browseForOpen()` e `browseForOpenMultiple()`. A caixa de diálogo browseForSave() também exibe somente arquivos de mídia, embora o usuário possa digitar um nome de arquivo arbitrariamente. Para abrir e salvar arquivos que não sejam de mídia, considere o uso de caixas de diálogo personalizadas em vez desses métodos.*

Modificação de caminhos do File

Adobe AIR 1.0 e posterior

Você também pode modificar o caminho de um objeto File existente, chamando o método `resolvePath()` ou modificando a propriedade `nativePath` ou `url` do objeto, conforme nos exemplos a seguir (no 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
```

Ao usar a propriedade `nativePath`, use o caractere de barra invertida (/) como caractere de separação de diretório. No Windows, você também pode usar o caractere de barra invertida (\), mas você não deve fazer isso, pois ele leva a um código que não funciona em várias plataformas.

Esquemas URL do AIR com suporte

Adobe AIR 1.0 e posterior

No AIR, você pode usar qualquer um dos seguintes esquemas URL na definição da propriedade `url` de um objeto File:

esquema de URL	Descrição
arquivo	Use para especificar um caminho relativo à raiz do sistema de arquivos. Por exemplo: `file:///c:/AIR Test/test.txt` A URL padrão especifica se a URL de arquivo usa a forma `file://<host>/<path>`. Como um caso especial, `<host>` pode ser a string vazia, que é interpretada como "a máquina a partir da qual a URL está sendo interpretada." Por essa razão, as URLs de arquivo normalmente têm três barras (///).
app	Use para especificar um caminho relativo ao diretório raiz do aplicativo instalado (o diretório que contém o arquivo application.xml do aplicativo instalado). Por exemplo, o caminho a seguir aponta para um subdiretório de imagens do diretório do aplicativo instalado: `app:/images`
app-storage	Use para especificar um caminho relativo ao diretório de armazenamento do aplicativo. Para cada aplicativo instalado, o AIR define um diretório exclusivo de armazenamento do aplicativo, que é um local útil para armazenar dados específicos desse aplicativo. Por exemplo, o caminho a seguir aponta para o arquivo prefs.xml em um subdiretório de configurações do diretório de armazenamento do aplicativo: `app-storage:/settings/prefs.xml`

Controle do backup de arquivos e do armazenamento em cache

Adobe AIR 3.6 e posterior, somente iOS e OS X

Determinados sistemas operacionais, sobretudo o iOS e o Mac OS X, oferecem aos usuários a capacidade de fazer backup de arquivos de aplicativo automaticamente para um armazenamento remoto. Além disso, há restrições no iOS no que diz respeito à possibilidade de os arquivos serem copiados ou não e ao local em que arquivos de diferentes propósitos podem ser armazenados.

A seção a seguir resume como cumprir as diretrizes da Apple para backup e armazenamento de arquivos. Para obter mais informações, consulte as próximas seções.

- Para determinar se um arquivo não precisa de backup e (somente para iOS) pode ser excluído pelo sistema operacional caso o espaço de armazenamento do dispositivo seja insuficiente, salve o arquivo no diretório do cache (`File.cacheDirectory`). Esse é o local de armazenamento preferencial no iOS e deve ser utilizado para a maioria dos arquivos que podem ser restaurados ou baixados novamente.
- Para determinar que um arquivo, embora não precise de backup, não deve ser excluído pelo sistema operacional, salve o arquivo em um dos diretórios da biblioteca de aplicativos, como o diretório de armazenamento de aplicativos (`File.applicationStorageDirectory`) ou o diretório de documentos (`File.documentsDirectory`). Defina a propriedade `preventBackup` do objeto File como `true`. Trata-se de uma exigência da Apple para conteúdos que possam ser restaurados ou novamente baixados, mas que constituem um requisito para o funcionamento adequado de seu aplicativo durante o uso offline.

Especificação de arquivos para backup

Com o objetivo de economizar o espaço de backup e diminuir o uso da largura de banda da rede, as diretrizes da Apple para aplicativos Mac e iOS determinam que somente arquivos que contenham dados inseridos por usuários ou dados que não possam ser restaurados ou baixados novamente devem ser destinados a backup.

Por padrão, todos os arquivos nas pastas da biblioteca de aplicativos recebem backup. No Mac OS X, trata-se do diretório de armazenamento de aplicativos. No iOS, isso inclui o diretório de armazenamento de aplicativos, o diretório de aplicativos, o diretório do desktop, o diretório de documentos e o diretório do usuário (visto que esses diretórios são mapeados para as pastas da biblioteca do aplicativo no iOS). Por conseguinte, todos os arquivos nestes diretórios são copiados para o armazenamento do servidor por padrão.

Se você estiver salvando um arquivo em um desses locais que possam ser recriados por seu aplicativo, deverá sinalizá-lo de modo que o sistema operacional saiba que não deverá fazer backup dele. Para indicar que o backup de um arquivo não deve ser feito, defina a propriedade `preventBackup` do objeto File como `true`.

Observe que no iOS, ao se tratar de um arquivo em qualquer pasta da biblioteca de aplicativos, mesmo se a propriedade `preventBackup` do arquivo estiver definida como `true`, o arquivo é sinalizado como um arquivo permanente que não deve ser excluído pelo sistema operacional.

Controle da exclusão e do armazenamento em cache do arquivo

As diretrizes da Apple para aplicativos iOS determina que, sempre que possível, o conteúdo que pode ser restaurado deve ser disponibilizado para ser excluído pelo sistema operacional caso não haja espaço de armazenamento suficiente no dispositivo.

No iOS, os arquivos nas pastas da biblioteca de aplicativos (como o diretório de armazenamento de aplicativos ou o diretório de documentos) são sinalizados como permanentes e não são excluídos pelo sistema operacional.

Salve os arquivos que possam ser restaurados pelo aplicativo e que não causarão problemas caso sejam excluídos em caso de espaço de armazenamento insuficiente no diretório do cache do aplicativo. Você poderá acessar o diretório do cache por meio da propriedade estática `File.cacheDirectory`.

No iOS, o diretório do cache corresponde ao diretório do cache do aplicativo (<Application Home>/Library/Caches). Em outros sistemas operacionais, esse diretório é mapeado para um diretório similar. Por exemplo, no Mac OS X, ele também mapeia para o diretório Caches na biblioteca de aplicativos. No Android, o diretório do cache mapeia para o diretório do cache do aplicativo. No Windows, o diretório do cache mapeia o diretório temporário do sistema operacional. Tanto no Android como no Windows, trata-se do mesmo diretório que é acessado por uma chamada para os métodos `createTempDirectory()` e `createTempFile()` da classe File.

Localização do caminho relativo entre dois arquivos

Adobe AIR 1.0 e posterior

Você pode usar o método `getRelativePath()` para localizar o caminho relativo entre dois arquivos:

```
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
```

O segundo parâmetro do método `getRelativePath()`, o parâmetro `useDotDot`, permite que a sintaxe `....` seja retornada nos resultados, para indicar diretórios pai:

```
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
```

Obtenção de versões canônicas de nomes de arquivo

Adobe AIR 1.0 e posterior

No Windows e no Mac OS, os nomes e caminhos de arquivo não fazem distinção entre letras maiúsculas e minúsculas. No exemplo a seguir, dois objetos File apontam para o mesmo arquivo:

```
File.documentsDirectory.resolvePath("test.txt");
File.documentsDirectory.resolvePath("TeSt.TxT");
```

No entanto, nomes de documentos e de diretórios incluem o uso de maiúsculas e minúsculas. Por exemplo, o seguinte assume que há uma pasta com nome AIR Test no diretório de documentos, como nos exemplos a seguir:

```
var file:File = File.documentsDirectory.resolvePath("AIR test");
trace(file.nativePath); // ... AIR test
file.canonicalize();
trace(file.nativePath); // ... AIR Test
```

O método `canonicalize()` converte o objeto `nativePath` para usar maiúsculas e minúsculas corretas para o nome de arquivo ou de diretório. Em sistemas de arquivo que fazem distinção entre maiúsculas e minúsculas (como Linux), quando vários arquivos possuem nomes cuja distinção é somente a letra maiúscula ou minúscula, o método `canonicalize()` ajusta o caminho para que corresponda ao primeiro arquivo encontrado (em uma ordem determinada pelo sistema de arquivos).

Você também pode usar o método `canonicalize()` para converter nomes de arquivos pequenos (nomes "8,3") em nomes de arquivos longos do Windows, como nos exemplos seguintes:

```
var path:File = new File();
path.nativePath = "C:\\AIR~1";
path.canonicalize();
trace(path.nativePath); // C:\AIR Test
```

Trabalho com pacotes e links simbólicos

Adobe AIR 1.0 e posterior

Vários sistemas operacionais oferecem suporte a arquivos de pacote e arquivos de link simbólico:

Pacotes – No Mac OS, os diretórios podem ser designados como pacotes e mostrados no Finder do Mac OS como um arquivo único, em vez de um diretório.

Links simbólicos — O Mac OS, o Linux e o Windows Vista oferecem suporte a links simbólicos. Os links simbólicos permitem que um arquivo aponte para outro arquivo ou diretório no disco. Embora semelhantes, os links simbólicos não são o mesmo que alias. O alias é sempre reportado como arquivo (em vez de diretório), e ler ou gravar em um alias ou atalho nunca afeta o diretório ou arquivo original para o qual ele aponta. Por outro lado, o link simbólico se comporta exatamente como o arquivo ou diretório para o qual aponta. Ele pode ser reportado como um arquivo ou diretório, e ler ou gravar em um link simbólico afetará o arquivo ou diretório para o qual ele aponta, não o link simbólico propriamente dito. Além disso, no Windows, a propriedade `isSymbolicLink` de um objeto File que faça referência a um ponto de junção (usado no sistema de arquivos NTFS) é definida como `true`.

A classe File inclui as propriedades `isPackage` e `isSymbolicLink` para verificar se o objeto File faz referência a um pacote ou link simbólico.

O código a seguir percorre o diretório da área de trabalho do usuário, listando subdiretórios que *não* sejam pacotes:

```
 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);
    }
}
```

O código a seguir percorre o diretório da área de trabalho do usuário, listando arquivos e diretórios que *não* sejam links simbólicos:

```
 var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (!desktopNodes[i].isSymbolicLink)
    {
        trace(desktopNodes[i].name);
    }
}
```

O método `canonicalize()` altera o caminho do link simbólico para apontar para o arquivo ou diretório ao qual o link se refere. O código a seguir percorre o diretório da área de trabalho do usuário e relata os caminhos referenciados pelos arquivos que sejam links simbólicos:

```
 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);
    }
}
```

Determinação de espaço disponível em um volume

Adobe AIR 1.0 e posterior

A propriedade `spaceAvailable` do objeto File é o espaço disponível para uso no local do File, em bytes. Por exemplo, o código a seguir verifica o espaço disponível no diretório de armazenamento do aplicativo:

```
trace(File.applicationStorageDirectory.spaceAvailable);
```

Se o objeto File fizer referência a um diretório, a propriedade `spaceAvailable` indicará o espaço no diretório que os arquivos podem usar. Se o objeto File fizer referência a um arquivo, a propriedade `spaceAvailable` indicará o espaço no qual o arquivo poderá crescer. Se o local do arquivo não existir, a propriedade `spaceAvailable` será definida como 0. Se o objeto File fizer referência a um link simbólico, a propriedade `spaceAvailable` será definida como espaço disponível no local para o qual o link simbólico aponta.

Geralmente o espaço disponível para um diretório ou arquivo é igual ao espaço disponível no volume que contém o diretório ou o arquivo. No entanto, o espaço disponível pode levar em conta limites por cotas e por diretório.

A adição de um arquivo ou diretório a um volume geralmente requer mais espaço que o tamanho real do arquivo ou o tamanho do conteúdo do diretório. Por exemplo, o sistema operacional pode exigir mais espaço para armazenar informações sobre índice. Ou os setores de disco necessários podem usar espaço adicional. Além disso, o espaço disponível muda dinamicamente. Dessa forma, você não poderá alocar todo o espaço informado para o armazenamento de arquivos. Para obter informações sobre como gravar no sistema de arquivos, consulte"Leitura e gravação de arquivos" na página 712.

O método `StorageVolumeInfo.getStorageVolumes()` fornece informações mais detalhadas sobre os volumes montados de armazenamento (consulte "Trabalho com volumes de armazenamento" na página 710).

Abertura de arquivos com o aplicativo padrão do sistema

Adobe AIR 2 e posterior

No AIR 2, você pode abrir um arquivo usando o aplicativo registrado pelo sistema operacional para abri-lo. Por exemplo, um aplicativo do AIR pode abrir um arquivo DOC com o aplicativo registrado para abri-lo. Utilize o método `openWithDefaultApplication()` de um objeto File para abrir o arquivo. Por exemplo, o código a seguir abre um arquivo chamado test.doc no computador do usuário e abre-o com o aplicativo padrão para os arquivos DOC:

```
var file:File = File.deskopDirectory;
file = file.resolvePath("test.doc");
file.openWithDefaultApplication();
```

Nota: *No Linux, o tipo MIME do arquivo, não a extensão do nome de arquivo, determina o aplicativo padrão para um arquivo.*

O código a seguir permite que o usuário navegue até um arquivo mp3 e abra-o no aplicativo padrão para reproduzir arquivos mp3:

```
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();
}
```

Você não pode usar o método `openWithDefaultApplication()` com arquivos localizados no diretório do aplicativo.

O AIR impede que você use o método `openWithDefaultApplication()` para abrir certos arquivos. No Windows, o AIR impede que você abra os arquivos que têm certos tipos de arquivos, como EXE ou BAT. No Mac OS, o AIR impede que você abra arquivos que serão lançados em certos aplicativos. (Incluindo Terminal e AppletLauncher no Mac OS, e csh, bash ou ruby no Linux). Tentar abrir um desses arquivos que usam o método `openWithDefaultApplication()` resulta em uma exceção. Para ver uma lista completa dos tipos de arquivo impedidos, consulte a entrada de referência de linguagem para o método `File.openWithDefaultApplication()`.

Nota: *Essa limitação não existe para um aplicativo do AIR instalado com um instalador nativo (um aplicativo de desktop expandido).*

Obtenção de informações sobre o sistema de arquivos

Adobe AIR 1.0 e posterior

A classe File inclui as seguintes propriedades estáticas que apresentam algumas informações úteis sobre o sistema de arquivos:

Propriedade	Descrição
`File.lineEnding`	A sequência de caracteres de final de linha usada pelo sistema operacional de hospedagem. No Mac OS e no Linux, ele é o caractere de alimentação de linha. No Windows, esse é o caractere de retorno de carro seguido pelo caractere de alimentação de linha.
`File.separator`	O caractere separador de componente do caminho do sistema operacional de hospedagem. No Mac OS e no Linux, ele é o caractere de barra (/). No Windows, ele é o caractere de barra invertida (\).
`File.systemCharset`	A codificação padrão usada em arquivos pelo sistema operacional de hospedagem. Ela pertence ao conjunto de caracteres usado pelo sistema operacional, correspondente ao respectivo idioma.

A classe `Capabilities` também inclui informações úteis sobre o sistema, que podem ser valiosas ao trabalhar com arquivos:

Propriedade	Descrição
Capabilities.hasIME	Especifica se o player está sendo executado em um sistema que possui (`true`) ou não possui (`false`) um IME (editor de métodos de entrada) instalado.
Capabilities.language	Especifica o código de idioma do sistema no qual o player está sendo executado.
Capabilities.os	Especifica o sistema operacional atual.

Nota: *Tenha cuidado ao usar `Capabilities.os` para determinar as características do sistema. Se existir uma propriedade mais específica para determinar a característica de um sistema, use-a. Do contrário, você corre o risco de gravar código que não funciona corretamente em todas as plataformas. Por exemplo, considere o seguinte código:*

```
var separator:String;
if (Capablities.os.indexOf("Mac") > -1)
{
    separator = "/";
}
else
{
    separator = "\\";
}
```

Esse código leva a problemas no Linux. É melhor usar simplesmente a propriedade `File.separator`.

Trabalho com diretórios

Adobe AIR 1.0 e posterior

O tempo de execução oferece a você recursos para trabalhar com diretórios no sistema de arquivos local.

Para obter detalhes sobre a criação de objetos File que apontam para diretórios, consulte "Como apontar um objeto File para um diretório" na página 693.

Criação de diretórios

Adobe AIR 1.0 e posterior

O método `File.createDirectory()` permite criar um diretório. Por exemplo, o código a seguir cria um diretório com nome AIR Test como um subdiretório do diretório inicial do usuário:

```
var dir:File = File.userDirectory.resolvePath("AIR Test");
dir.createDirectory();
```

Se o diretório existir, o método `createDirectory()` não faz nada.

Além disso, em alguns modos, o objeto FileStream cria diretórios ao abrir arquivos. Diretórios ausentes são criados quando você cria uma ocorrência FileStream com o parâmetro `fileMode` do construtor `FileStream()` definido como `FileMode.APPEND` ou `FileMode.WRITE`. Para obter mais informações, consulte "Fluxo de trabalho de leitura e gravação de arquivos" na página 712.

Criação de diretório temporário

Adobe AIR 1.0 e posterior

A classe File inclui o método `createTempDirectory()`, que cria um diretório na pasta de diretórios temporários do Sistema, como no exemplo a seguir:

```
var temp:File = File.createTempDirectory();
```

O método `createTempDirectory()` cria automaticamente um diretório temporário exclusivo (poupando o seu trabalho de determinar um novo local exclusivo).

Você pode usar um diretório temporário para armazenar temporariamente arquivos temporários usados em uma sessão do aplicativo. Observe que há um método `createTempFile()` para criar arquivos temporários novos e exclusivos no diretório temporário System.

Pode ser conveniente excluir o diretório temporário antes de fechar o aplicativo, uma vez que ele *não* é excluído automaticamente em todos os dispositivos.

Enumeração de diretórios

Adobe AIR 1.0 e posterior

Você pode usar o método `getDirectoryListing()` ou o método `getDirectoryListingAsync()` do objeto File para obter uma matriz de objetos File apontando para arquivos e subpastas em um diretório.

Por exemplo, o código a seguir lista o conteúdo do diretório de documentos do usuário (sem examinar subdiretórios):

```
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);
}
```

Ao usar a versão assíncrona do método, o objeto de evento `directoryListing` terá a propriedade `files` que é a matriz de objetos File pertencentes aos diretórios:

```
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);
    }
}
```

Cópia e movimentação de diretórios

Adobe AIR 1.0 e posterior

Você pode copiar ou mover o diretório, usando os mesmos métodos utilizados para copiar ou mover um arquivo. Por exemplo, o código a seguir copia um diretório de forma síncrona:

```
var sourceDir:File = File.documentsDirectory.resolvePath("AIR Test");
var resultDir:File = File.documentsDirectory.resolvePath("AIR Test Copy");
sourceDir.copyTo(resultDir);
```

Quando você especifica true para o parâmetro `overwrite` do método `copyTo()`, todos arquivos e pastas em um diretório de destino existente são excluídos e substituídos pelos arquivos e pastas no diretório de origem (mesmo se o arquivo de destino não existir no diretório de origem).

O diretório especificado como o parâmetro `newLocation` do método `copyTo()` especifica o caminho para o diretório resultante; ele *não* especifica o diretório *pai* que conterá o diretório resultante.

Para obter detalhes, consulte "Cópia e movimentação de arquivos" na página 708.

Exclusão de conteúdo do diretório

Adobe AIR 1.0 e posterior

A classe File inclui os métodos `deleteDirectory()` e `deleteDirectoryAsync()`. Esses métodos excluem diretórios; o primeiro trabalha de forma síncrona, o segundo trabalha de forma assíncrona (consulte "Noções básicas do arquivo AIR" na página 689). Os dois métodos incluem o parâmetro `deleteDirectoryContents` (com um valor Booliano); quando esse parâmetro estiver definido como `true` (o valor padrão será `false`) a chamada do método excluirá diretórios não vazios; caso contrário, apenas diretórios vazios serão excluídos.

Por exemplo, o código a seguir exclui de modo síncrono o subdiretório AIR Test do diretório de documentos do usuários:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.deleteDirectory(true);
```

O código a seguir exclui de modo assíncrono o subdiretório AIR Test do diretório de documentos do usuário:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.addEventListener(Event.COMPLETE, completeHandler)
directory.deleteDirectoryAsync(true);

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Também estão incluídos os métodos `moveToTrash()` e `moveToTrashAsync()`, que você pode usar para mover um diretório para a lixeira do Sistema. Para obter detalhes, consulte "Movimentação de arquivo para a lixeira" na página 710.

Trabalho com arquivos

Adobe AIR 1.0 e posterior

Usando a API de arquivo AIR, você pode adicionar recursos básicos de interação de arquivo a seus aplicativos. Por exemplo, você pode ler e gravar arquivos, copiar e excluir arquivos e assim por diante. Como seus aplicativos podem acessar o sistema de arquivos local, consulte "Segurança do AIR" na página 1116, se ainda não tiver feito isso.

Nota: *Você pode associar um tipo de arquivo a um aplicativo do AIR (de modo que, quando você clicar nele duas vezes, o aplicativo seja aberto). Para obter detalhes, consulte "Gerenciamento de associações de arquivos" na página 916.*

Obtenção de informações de arquivos

Adobe AIR 1.0 e posterior

A classe File inclui as seguintes propriedades que oferecem informações sobre um arquivo ou diretório para o qual o objeto File aponta:

Propriedade File	Descrição
creationDate	A data de criação do arquivo no disco local.
creator	Obsoleto — usa a propriedade `extension`. (Essa propriedade relata o tipo de criador Macintosh do arquivo, usado somente nas versões do Mac OS anteriores ao Mac OS X).
downloaded	(No Air 2 e posteriores) Indica se o arquivo ou o diretório mencionados foram baixados (da Internet) ou não. Esta propriedade só é significativa em sistemas operacionais em que os arquivos podem ser marcados como baixados: • Windows XP Service Pack 2 e versões posteriores e no Windows Vista • SO Mac 10.5 e posterior
exists	Se o arquivo ou diretório referenciado existir.
extension	A extensão de arquivo, que é a parte seguinte ao nome (e não incluindo) o ponto final ("."). Se não houver ponto no nome do arquivo, a extensão será `null`.
ícone	Um objeto Icon que contém os ícones definidos para o arquivo.
isDirectory	Se a referência do objeto File for a um diretório.
modificationDate	A data da última modificação do arquivo ou diretório no disco local.
nome	O nome do arquivo ou diretório (incluindo a extensão de arquivo, se houver) no disco local.
nativePath	O caminho completo na representação do sistema operacional de hospedagem. Consulte "Caminhos de objetos File" na página 690.
parent	A pasta que contém a pasta ou o arquivo representado pelo objeto File. Essa propriedade será `null` se o objeto File fizer referência a um arquivo ou diretório na raiz do sistema de arquivos.
size	O tamanho do arquivo no disco local, em bytes.
tipo	Obsoleto — usa a propriedade `extension`. (No Macintosh, essa propriedade é o tipo de arquivo com quatro caracteres, usado somente nas versões do Mac OS anteriores ao Mac OS X).
url	A URL do arquivo ou diretório. Consulte "Caminhos de objetos File" na página 690.

Para obter detalhes dessas propriedades, consulte a entrada da classe File na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Cópia e movimentação de arquivos

Adobe AIR 1.0 e posterior

A classe File inclui dois métodos de cópia de arquivos ou diretórios: `copyTo()` e `copyToAsync()`. A classe File inclui dois métodos para mover arquivos ou diretórios: `moveTo()` e `moveToAsync()`. Os métodos `copyTo()` e `moveTo()` funcionam de modo síncrono e os métodos `copyToAsync()` e `moveToAsync()` funcionam de modo assíncrono (consulte "Noções básicas do arquivo AIR" na página 689).

Para copiar ou mover um arquivo, defina dois objetos File. Um aponta para o arquivo que deve ser copiado ou movido, e é o objeto que chama o método de cópia ou movimentação; o outro aponta para o caminho de destino (resultado).

O seguinte copia o arquivo test.txt do subdiretório AIR Test do diretório de documentos do usuário em um arquivo com nome copy.txt no mesmo diretório:

```
 var original:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var newFile:File = File.resolvePath("AIR Test/copy.txt");
original.copyTo(newFile, true);
```

Nesse exemplo, o valor do parâmetro `overwrite` do método `copyTo()` (o segundo parâmetro) é definido como `true`. Definindo `overwrite` como `true`, o arquivo de destino existente é sobrescrito. Esse parâmetro é opcional. Se você defini-lo como `false` (o valor padrão), a operação despacha um evento IOErrorEvent se o arquivo de destino existir (e o arquivo não é copiado).

As versões "Async" dos métodos de cópia e movimentação funcionam de modo assíncrono. Use o método `addEventListener()` para monitorar a conclusão da tarefa ou as condições de erro, como no código a seguir:

```
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.");
}
```

A classe File também inclui os métodos `File.moveToTrash()` e `File.moveToTrashAsync()`, que movem o arquivo ou diretório para a lixeira do sistema.

Exclusão de arquivo

Adobe AIR 1.0 e posterior

A classe File inclui os métodos `deleteFile()` e `deleteFileAsync()`. Esses métodos excluem arquivos; o primeiro trabalha de forma síncrona, o segundo trabalha de forma assíncrona (consulte "Noções básicas do arquivo AIR" na página 689).

Por exemplo, o código a seguir exclui de modo síncrono o arquivo test.txt do diretório de documentos do usuário:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.deleteFile();
```

O código a seguir exclui de modo assíncrono o arquivo test.txt do diretório de documentos do usuário:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, completeHandler)
file.deleteFileAsync();

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Também estão incluídos os métodos `moveToTrash()` e `moveToTrashAsync()`, que você pode usar para mover um arquivo ou diretório para a lixeira do Sistema. Para obter detalhes, consulte "Movimentação de arquivo para a lixeira" na página 710.

Movimentação de arquivo para a lixeira

Adobe AIR 1.0 e posterior

A classe File inclui os métodos `moveToTrash()` e `moveToTrashAsync()`. Esses métodos enviam um arquivo ou diretório para a lixeira do Sistema; o primeiro trabalha de forma síncrona, o segundo trabalha de forma assíncrona (consulte "Noções básicas do arquivo AIR" na página 689).

Por exemplo, o código a seguir move de modo síncrono o arquivo test.txt no diretório de documentos do usuários para a lixeira do Sistema:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.moveToTrash();
```

Nota: *Em sistemas operacionais que não oferecem suporte ao conceito de uma pasta de lixeira recuperável, os arquivos são removidos imediatamente.*

Criação de arquivo temporário

Adobe AIR 1.0 e posterior

A classe File inclui o método `createTempFile()`, que cria um arquivo na pasta de diretórios temporários do Sistema, como no exemplo a seguir:

```
var temp:File = File.createTempFile();
```

O método `createTempFile()` cria automaticamente um arquivo temporário exclusivo (poupando o seu trabalho de determinar um novo local exclusivo).

Você pode usar um arquivo temporário para armazenar temporariamente informações usadas em uma sessão do aplicativo. Observe que também há o método `createTempDirectory()` para criar um diretório temporário exclusivo no diretório temporário System.

Pode ser conveniente excluir o arquivo temporário antes de fechar o aplicativo, uma vez que ele *não* é excluído automaticamente em todos os dispositivos.

Trabalho com volumes de armazenamento

Adobe AIR 2 e posterior

No AIR 2, você pode detectar quando os volumes de armazenamento em massa são montados ou desmontados. A classe StorageVolumeInfo define um objeto `storageVolumeInfo` singleton. O objeto `StorageVolumeInfo.storageVolumeInfo` despacha um evento `storageVolumeMount` quando um volume de armazenamento é montado. E também despacha um evento `storageVolumeUnmount` quando um volume é desmontado. A classe StorageVolumeChangeEvent define esses eventos.

Nota: *Em distribuições modernas de Linux, o objeto StorageVolumeInfo envia apenas eventos `storageVolumeMount` and `storageVolumeUnmount` para dispositivos físicos e unidades de rede montados em locais específicos.*

A propriedade `storageVolume` da classe StorageVolumeChangeEvent é um objeto StorageVolume. A classe StorageVolume define as propriedades básicas do volume de armazenamento:

- `unidade`—A letra da unidade do volume no Windows (`null` em outros sistemas operacionais)
- `fileSystemType` — O tipo do sistema de arquivo no volume de armazenamento (como "FAT", "NTFS", "HFS" ou "UFS")

- `isRemoveable` — Se um volume é removível `true`) ou não (`false`)
- `isWritable` — Se um volume é gravável `true`) ou não (`false`)
- `name` — O nome do volume
- `rootDirectory` — Um objeto de arquivo que corresponde ao diretório raiz do volume

A classe StorageVolumeChangeEvent também contém uma propriedade `rootDirectory`. A propriedade `rootDirectory` é um objeto File que faz referência ao diretório raiz do volume de armazenamento que foi montado ou desmontado.

A propriedade `storageVolume` do objeto StorageVolumeChangeEvent não está definida (`null`) para o volume não montado. No entanto, você pode acessar a propriedade `rootDirectory` do evento.

O código a seguir libera o nome e o caminho do arquivo de um volume de armazenamento quando é montado:

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_MOUNT, onVolumeMount);
function onVolumeMount(event:StorageVolumeChangeEvent):void
{
    trace(event.storageVolume.name, event.rootDirectory.nativePath);
}
```

O código a seguir libera o caminho do arquivo de um volume de armazenamento quando é desmontado:

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_UNMOUNT, onVolumeUnmount);
function onVolumeUnmount(event:StorageVolumeChangeEvent):void
{
    trace(event.rootDirectory.nativePath);
}
```

O objeto `StorageVolumeInfo.storageVolumeInfo` inclui um método `getStorageVolumes()`. Esse método retorna um vetor dos objetos StorageVolume que correspondem aos volumes de armazenamento atualmente montados. O código a seguir mostra como listar os nomes e os diretórios raiz de todos os volumes de armazenamento montados:

```
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);
}
```

Nota: *Em distribuições modernas de Linux, o método `getStorageVolumes()` retorna objetos que correspondem a dispositivos físicos e unidades de rede montados em locais específicos.*

O método `File.getRootDirectories()` lista os diretórios raiz (consulte "Apontando para a raiz do sistema de arquivos" na página 695. No entanto, os objetos StorageVolume (enumerados pelo método `StorageVolumeInfo.getStorageVolumes()`) fornece mais informações sobre os volumes de armazenamento.

Você pode usar a propriedade `spaceAvailable` da propriedade `rootDirectory` de um objeto StorageVolume para obter o espaço disponível em um volume de armazenamento. (Consulte "Determinação de espaço disponível em um volume" na página 703.)

Mais tópicos da Ajuda

StorageVolume

StorageVolumeInfo

Leitura e gravação de arquivos

Adobe AIR 1.0 e posterior

A classe FileStream permite que aplicativos AIR leiam e gravem no sistema de arquivos.

Fluxo de trabalho de leitura e gravação de arquivos

Adobe AIR 1.0 e posterior

O fluxo de trabalho de leitura e gravação de arquivos é o seguinte.

Inicializa o objeto File que aponta para o caminho.

O objeto File representa o caminho do arquivo com que você deseja trabalhar (ou um arquivo que você criará mais tarde).

```
var file:File = File.documentsDirectory;
file = file.resolvePath("AIR Test/testFile.txt");
```

Este exemplo usa a propriedade `File.documentsDirectory` e o método `resolvePath()` do objeto File para inicializá-lo. No entanto, há várias outras maneiras de apontar o objeto File para um arquivo. Para obter mais informações, consulte "Como apontar um objeto File para um arquivo" na página 697.

Inicializa o objeto FileStream.

Chama o método open() ou openAsync() do objeto FileStream.

O método que você chama depende de você desejar abrir o arquivo para operações síncronas ou assíncronas. Use o objeto File como parâmetro `file` do método open. No parâmetro `fileMode`, especifique uma constante da classe FileMode que determine o modo como você usará o arquivo.

Por exemplo, o código a seguir inicializa o objeto FileStream que é usado para criar um arquivo e substituir algum dado existente:

```
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);
```

Para obter mais informações, consulte"Inicialização de um objeto FileStream e abertura e fechamento de arquivos" na página 714, e "modos de abertura FileStream" na página 713.

Se você abriu o arquivo de modo assíncrono (usando o método openAsync()), adicione e configure ouvintes de eventos para o objeto FileStream.

Esses métodos ouvintes de eventos respondem a eventos gerados pelo objeto FileStream em diversas situações. Essas situações incluem quando os dados são lidos do arquivo, quando são encontrados erros de E/S, ou quando a quantidade completa de dados a serem gravados foi gravada.

Para obter detalhes, consulte "Programação assíncrona e eventos gerados por um objeto FileStream aberto assincronicamente" na página 718.

Incluir código de leitura e gravação de dados, conforme necessário.

Há vários métodos da classe FileStream relacionados a leitura e gravação. (Cada um deles inicia com "read" ou "write"). O método escolhido para ser usado na leitura ou gravação de dados depende do formato de dados no arquivo de destino.

Por exemplo, se os dados no arquivo de destino forem texto de codificação UTF, você pode usar os métodos `readUTFBytes()` e `writeUTFBytes()`. Se desejar tratar os dados como matrizes de bytes, você pode usar os métodos `readByte()`, `readBytes()`, `writeByte()` e `writeBytes()`. Para saber detalhes, consulte "Formatos de dados e seleção de métodos de leitura e gravação para uso" na página 719.

Se você abriu o arquivo de modo assíncrono, certifique-se de que há dados suficientes disponíveis antes de chamar o método de leitura. Para obter detalhes, consulte "O buffer de leitura e a propriedade bytesAvailable do objeto FileStream" na página 716.

Antes de fazer a gravação em um arquivo, se desejar verificar a quantidade de espaço em disco disponível, você poderá assinalar a propriedade spaceAvailable do objeto File. Para obter mais informações, consulte "Determinação de espaço disponível em um volume" na página 703.

Chama o método close() do objeto FileStream quando você tiver terminado o trabalho no arquivo.
Chamar o método close() disponibiliza o arquivo para outros aplicativos.

Para obter detalhes, consulte "Inicialização de um objeto FileStream e abertura e fechamento de arquivos" na página 714.

Para visualizar um aplicativo de amostra que usa a classe FileStream para ler e gravar arquivos, consulte os seguintes artigos no Centro de desenvolvimento do Adobe AIR:

- Criação de editor de arquivo de texto
- Criação de editor de arquivo de texto
- Leitura e gravação de um arquivo de preferências XML

Trabalho com objetos FileStream

Adobe AIR 1.0 e posterior

A classe FileStream define métodos de abertura, leitura e gravação de arquivos.

modos de abertura FileStream

Adobe AIR 1.0 e posterior

Cada um dos métodos `open()` e `openAsync()` do objeto FileStream inclui o parâmetro `fileMode`, que define algumas propriedades de um fluxo de arquivo, incluindo o seguinte:

- A capacidade de ler do arquivo
- A capacidade de gravar no arquivo
- Se os dados serão sempre acrescentados ao final do arquivo (durante a gravação).
- O que fazer quando o arquivo não existir (e quando os respectivos diretórios pai não existirem)

A seguir, os vários modos de arquivo (que você pode especificar como o parâmetro `fileMode` dos métodos `open()` e `openAsync()`):

modo File	Descrição
FileMode.READ	Especifica que o arquivo está aberto somente para leitura.
FileMode.WRITE	Especifica que o arquivo está aberto para gravação. Se o arquivo não existir, ele é criado quando o objeto FileStream é aberto. Se o arquivo existir, todos os dados existentes são excluídos.
FileMode.APPEND	Especifica que o arquivo está aberto para acréscimo. O arquivo é criado se ele não existir. Se o arquivo existir, os dados existentes não são substituídos e todas as gravações serão iniciadas no final do arquivo.
FileMode.UPDATE	Especifica que o arquivo está aberto para leitura e gravação. Se o arquivo não existir, ele é criado. Especifique esse modo para acesso de leitura/gravação aleatório ao arquivo. Você pode ler a partir de qualquer posição no arquivo. Ao gravar no arquivo, apenas os bytes gravados substituirão os bytes existentes (todos os demais bytes permanecerão inalterados).

Inicialização de um objeto FileStream e abertura e fechamento de arquivos

Adobe AIR 1.0 e posterior

Ao abrir o objeto FileStream, você o torna disponível para a leitura e gravação de dados em um arquivo. Você abre o objeto FileStream passando o objeto File para o método `open()` ou `openAsync()` do objeto FileStream:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
```

O parâmetro `fileMode` (o segundo parâmetro dos métodos `open()` e `openAsync()`) especifica o modo em que o arquivo deve ser aberto: para leitura, gravação, acréscimo ou atualização. Para obter detalhes, consulte a seção anterior,"modos de abertura FileStream" na página 713.

Se você usar o método `openAsync()` para abrir o arquivo para operações de arquivo de modo assíncrono, configure os ouvintes de eventos para tratar os eventos assíncronos:

```
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 {
    // ...
}
```

O arquivo é aberto para operações síncronas ou assíncronas, dependendo de você usar o método `open()` ou `openAsync()`. Para obter detalhes, consulte "Noções básicas do arquivo AIR" na página 689.

Se você definir o parâmetro `fileMode` como `FileMode.READ` ou `FileMode.UPDATE` no método de abertura do objeto FileStream, os dados são lidos no buffer de leitura tão logo você abra o objeto FileStream. Para obter detalhes, consulte "O buffer de leitura e a propriedade bytesAvailable do objeto FileStream" na página 716.

Você pode chamar o método `close()` do objeto FileStream para fechar o arquivo associado, tornando-o disponível para uso por outros aplicativos.

A propriedade position do objeto FileStream

Adobe AIR 1.0 e posterior

A propriedade `position` do objeto FileStream determina quando os dados são lidos ou gravados no próximo método de leitura ou gravação.

Antes de uma operação de leitura ou gravação, defina a propriedade `position` como qualquer posição válida no arquivo.

Por exemplo, o código a seguir grava a sequência `"hello"` (na codificação UTF) na posição 8 do arquivo:

```
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");
```

Quando você abre pela primeira vez o objeto FileStream, a propriedade `position` é definida como 0.

Antes de uma operação de leitura, o valor de `position` deve ser no mínimo 0 e inferior ao número de bytes no arquivo (que são posições existentes no arquivo).

O valor da propriedade `position` é modificado apenas nas seguintes condições:

- Quando você define explicitamente a propriedade `position`.
- Quando você chama o método de leitura.
- Quando você chama o método de gravação.

Quando você chama um método de leitura ou gravação do objeto FileStream, a propriedade `position` é imediatamente incrementada pelo número de bytes que você lê ou grava. Dependendo do método de leitura usado, a propriedade `position` também é incrementada pelo número de bytes especificado para leitura ou pelo número de bytes disponível. Quando você chama o método de leitura ou gravação subsequentemente, ele faz a leitura ou gravação iniciando na nova posição.

```
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
```

Há, porém, uma exceção: em um FileStream aberto no modo append, a propriedade `position` não é alterada após uma chamada de um método de gravação. (No modo append, os dados são sempre gravados no final do arquivo, independentemente do valor da propriedade `position`.)

Em um arquivo aberto para operações assíncronas, a operação de gravação não é concluída antes de a próxima linha de código ser executada. No entanto, você pode chamar vários métodos assíncronos sequencialmente e o tempo de execução os executa em ordem:

```
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.");
}
```

A saída trace desse código é a seguinte:

```
started.
finished.
```

Você *pode* especificar o valor `position` logo após chamar um método de leitura ou gravação (ou a qualquer momento) e a próxima operação de leitura ou gravação ocorrerá no início dessa posição. Por exemplo, observe que o código a seguir define a propriedade `position` logo após uma chamada para a operação `writeBytes()` e a `position` é definida como esse valor (300) mesmo após a operação de gravação ser concluída:

```
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
```

O buffer de leitura e a propriedade bytesAvailable do objeto FileStream

Adobe AIR 1.0 e posterior

Quando o objeto FileStream com recursos de leitura (em que o parâmetro `fileMode` do método `open()` ou `openAsync()` foi definido como `READ` ou `UPDATE`) for aberto, o tempo de execução armazena os dados em um buffer interno. O objeto FileStream começa a fazer a leitura de dados no buffer assim que o arquivo for aberto (chamando o método `open()` ou `openAsync()` do objeto FileStream).

Em um arquivo aberto para operações síncronas (usando o método `open()`), você sempre poderá definir o ponteiro `position` em qualquer posição válida (dentro dos limites do arquivo) e iniciar a leitura de qualquer quantidade de dados (dentro dos limites do arquivo), conforme mostra o código a seguir (que supõe que o arquivo contém pelo menos 100 bytes):

```
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);
```

Quer o arquivo seja aberto para operações síncronas ou assíncronas, os métodos de leitura sempre fazem a leitura dos bytes "disponíveis", representados pela propriedade `bytesAvalable`. Ao fazer a leitura sincronicamente, todos os bytes do arquivo ficam disponíveis o tempo todo. Ao fazer a leitura assincronicamente, os bytes se tornam disponíveis, iniciando na posição especificada pela propriedade `position`, em uma série de preenchimentos de buffer assíncronos assinalados por eventos `progress`.

Em arquivos abertos para operações *síncronas*, a propriedade `bytesAvailable` é sempre definida para representar o número de bytes da propriedade `position` no final do arquivo (todos os bytes no arquivo estarão sempre disponíveis para leitura).

Em arquivos abertos para operações *assíncronas*, é necessário assegurar que o buffer de leitura consumiu dados suficientes antes de chamar o método de leitura. Em um arquivo aberto assincronicamente, conforme o andamento da operação de leitura, os dados do arquivo, iniciando na `posição` especificada quando a operação de leitura iniciou, são adicionados ao buffer, e a propriedade `bytesAvailable` é incrementada com cada byte lido. A propriedade `bytesAvailable` indica o número de bytes disponíveis, iniciando com o byte na posição especificada pela propriedade `position` até o final do buffer. Periodicamente, o objeto FileStream envia um evento `progress`.

Em um arquivo aberto sincronicamente, à medida que os dados se tornam disponíveis no buffer de leitura, o objeto FileStream despacha periodicamente o evento `progress`. Por exemplo, o código a seguir lê dados em um objeto ByteArray, `bytes`, conforme é lido no buffer:

```
 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);
}
```

Em um arquivo aberto assincronicamente, apenas os dados no buffer de leitura podem ser lidos. Além disso, conforme você lê os dados, eles são removidos do buffer de leitura. Em operações de leitura você precisa assegurar que haja dados no buffer de leitura antes de chamar uma operação de leitura. Por exemplo, o código a seguir faz a leitura de 8000 bytes de dados iniciando da posição 4000 no arquivo:

```
 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");
    }
}
```

Durante a operação de gravação, o objeto FileStream não faz a leitura de dados no buffer de leitura. Quando a operação de gravação estiver concluída (todos os dados no buffer de gravação são gravados no arquivo), o objeto FileStream inicia um novo buffer de leitura (supondo que o objeto FileStream associado foi aberto com recursos de leitura) e inicia a leitura de dados no buffer de leitura, começando da posição especificada pela propriedade `position`. A propriedade `position` pode ser a posição do último byte gravado ou pode ser uma posição diferente, se o usuário especificar um valor diferente para o objeto `position` após a operação de gravação.

Programação assíncrona e eventos gerados por um objeto FileStream aberto assincronicamente

Adobe AIR 1.0 e posterior

Quando um arquivo for aberto de modo assíncrono (usando o método `openAsync()`), os arquivos de leitura e gravação serão executados de modo assíncrono. À medida que os dados são lidos no buffer de leitura e os dados de saída começam a ser gravados, outro código ActionScript pode ser executado.

Isso significa que é necessário se registrar para eventos gerados pelo objeto FileStream abertos de modo assíncrono.

Registrando-se para o evento `progress`, você poderá ser notificado quando novos dados se tornarem disponíveis para leitura, como no seguinte código:

```
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");
}
```

Você pode fazer a leitura de todos os dados se registrando para o evento `complete`, como no seguinte código:

```
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");
}
```

Da mesma maneira que os dados de entrada são colocados em buffer para permitir a leitura assíncrona, os dados que você grava em um fluxo assíncrono são colocados em buffer e gravados no arquivo de forma assíncrona. À medida que os dados são gravados em um arquivo, o objeto FileStream despacha periodicamente o objeto `OutputProgressEvent`. O objeto `OutputProgressEvent` inclui a propriedade `bytesPending` definida pelo número de bytes restantes para gravação. Você pode registrar para que o evento `outputProgress` seja notificado, já que esse buffer é, na realidade, gravado no arquivo, talvez para exibir uma caixa de diálogo de progresso. No entanto, isso geralmente não é necessário. Especificamente, você pode chamar o método `close()` sem se importar com os bytes não gravados. O objeto FileStream continuará gravando dados e o evento `close` será entregue após o byte final ser gravado no arquivo e o arquivo subjacente ser fechado.

Formatos de dados e seleção de métodos de leitura e gravação para uso

Adobe AIR 1.0 e posterior

Todo arquivo é um conjunto de bytes em um disco. No ActionScript, os dados de um arquivo sempre podem ser representados como ByteArray. Por exemplo, o código a seguir faz a leitura de dados de um arquivo em um objeto ByteArray chamado `bytes`:

```
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);
}
```

De forma semelhante, o código seguinte grava dados de um ByteArray chamado `bytes` em um arquivo:

```
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);
```

No entanto, normalmente você não quer armazenar dados em um objeto ByteArray do ActionScript. E, geralmente, o arquivo de dados está em um formato de arquivo especificado.

Por exemplo, os dados no arquivo podem estar em um formato de arquivo de texto e talvez você deseje representar esses dados em um objeto String.

Por esse motivo, a classe FileStream inclui métodos de leitura e gravação para leitura e gravação de dados em outros tipos diferentes dos objetos ByteArray e a partir deles. Por exemplo, o método `readMultiByte()` permite fazer a leitura de dados de um arquivo e armazená-los em uma sequência, como no código a seguir:

```
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");
}
```

O segundo parâmetro do método `readMultiByte()` especifica o formato de texto que o ActionScript usa para interpretar os dados ("iso-8859-1" no exemplo). O Adobe AIR permite o uso de codificações comuns de conjuntos de caracteres (consulte Conjuntos de caracteres permitidos).

A classe FileStream também inclui o método `readUTFBytes()`, que faz leitura de dados do buffer de leitura em uma sequência, usando o conjunto de caracteres UTF-8. Como os caracteres no conjunto de caracteres UTF-8 são de tamanho variável, não use `readUTFBytes()` em um método que responde ao evento `progress`, já que os dados no final do buffer de leitura podem representar um caractere incompleto. (Isso também é verdadeiro ao usar o método `readMultiByte()` com uma codificação de caractere de tamanho variável). Por esse motivo, leia o conjunto de dados inteiro quando o objeto FileStream despachar o evento `complete`.

Também há métodos de gravação semelhantes, `writeMultiByte()` e `writeUTFBytes()`, para trabalhar com objetos String e arquivos de texto.

Os métodos `readUTF()` e `writeUTF()` (não confundir com `readUTFBytes()` e `writeUTFBytes()`) também fazem leitura e gravação de dados de texto em um arquivo, mas eles supõem que os dados de texto são precedidos por dados que especificam o tamanho dos dados de texto, o que não é prática comum em arquivos de texto padrão.

Alguns arquivos de texto de codificação UTF começam com um caractere "UTF-BOM" (marca de ordem de bytes) que define que a terminação, bem como o formato de codificação (como UTF-16 ou UTF-32).

Para obter um exemplo de leitura e gravação de um arquivo de texto, consulte "Exemplo: Leitura de arquivo XML em um objeto XML" na página 721.

`readObject()` e `writeObject()` são formas convenientes de armazenar e recuperar dados de objetos ActionScript complexos. Os dados são codificados em AMF (ActionScript Message Format). O Adobe AIR, o Flash Player, o Flash Media Server e o Flex Data Services incluem APIs para trabalhar co dados neste formato.

Há mais alguns métodos de leitura e gravação (como`readDouble()` e `writeDouble()`). No entanto, se você usá-los, certifique-se de que o formato de arquivo corresponde ao formato dos dados definidos por esses métodos.

Os formatos de arquivos são normalmente mais complexos do que os formatos de texto simples. Por exemplo, o arquivo MP3 inclui dados compactados que só podem ser interpretados com a descompactação e os algoritmos de decodificação específicos para arquivos MP3. Arquivos MP3 também podem incluir marcas ID3 que contêm informações de metatag sobre o arquivo (como o título e o artista de uma canção). Há várias versões do formato ID3, mas a mais simples (ID3 versão 1) é discutida na seção "Exemplo: Leitura e gravação de dados com acesso aleatório" na página 722.

Outros formatos de arquivo (de imagens, bancos de dados, documentos de aplicativos etc.) têm estruturas diferentes e, para trabalhar com os respectivos dados no ActionScript, você deve saber como os dados são estruturados.

Uso dos métodos load() e save()

Flash Player 10 e posterior, Adobe AIR 1.5 e posterior

O Flash Player 10 adicionou os métodos `load()` e `save()` à classe FileReference. Esses métodos também estão no AIR 1.5 e a classe File herda os métodos da classe FileReference. Esses métodos foram projetados para fornecer um meio seguro para usuários carregarem e salvarem dados de arquivo em Flash Player. No entanto, os aplicativos AIR também podem usar esses métodos como uma maneira fácil de carregar e salvar arquivos de forma assíncrona.

Por exemplo, o código a seguir salva uma sequência de caracteres em um arquivo de texto:

```
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.");
}
```

O parâmetro `data` do método `save()` pode usar um valor String, XML ou ByteArray. Quando o argumento é um valor String ou XML, o método salva o arquivo como arquivo de texto criptografado UTF-8.

Quando esse exemplo de código é executado, o aplicativo exibe uma caixa de diálogo em que o usuário seleciona o destino do arquivo salvo.

O código a seguir carrega uma sequência de caracteres de um arquivo de texto codificado em UTF-8.

```
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);
}
```

A classe FileStream fornece mais funcionalidade do que os métodos `load()` e `save()`.

- Usando a classe FileStream, você pode ler e gravar dados de forma síncrona e assíncrona.
- O uso da classe FileStream permite gravar em um arquivo de forma incremental.
- O uso da classe FileStream permite abrir um arquivo para acesso aleatório (para leitura e gravação em qualquer seção do arquivo).
- A classe FileStream permite especificar o tipo de acesso que você tem ao arquivo, definindo o parâmetro `fileMode` do método `open()` ou `openAsync()`.
- A classe FileStream permite salvar dados em arquivos sem apresentar ao usuário uma caixa de diálogo Abrir ou Salvar.
- Você pode usar diretamente tipos diferentes de matrizes de bytes ao ler dados com a classe FileStream.

Exemplo: Leitura de arquivo XML em um objeto XML

Adobe AIR 1.0 e posterior

O exemplo a seguir demonstra como fazer a leitura e a gravação de um arquivo de texto contendo dados XML.

Para ler o arquivo, inicialize os objetos File e FileStream, chame o método `readUTFBytes()` do FileStream e converta a sequência em um objeto XML:

```
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();
```

De modo semelhante, gravar dados no arquivo é tão fácil quanto configurar os objetos File e FileStream apropriados e, em seguida, chamar um método de gravação do objeto FileStream. Passe a versão de sequência dos dados XML para o método de gravação, como no código seguinte:

```
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();
```

Esses exemplos usam os métodos `readUTFBytes()` e `writeUTFBytes()`, porque consideram que os arquivos estejam no formato UTF-8. Caso contrário, você pode precisar usar um método diferente (consulte "Formatos de dados e seleção de métodos de leitura e gravação para uso" na página 719).

Os exemplos anteriores usam objetos FileStream abertos para operação síncrona. Você também pode abrir arquivos para operações assíncronas (que dependem das funções de ouvinte de evento para responder aos eventos). Por exemplo, o código a seguir mostra como fazer a leitura de um arquivo XML de modo assíncrono:

```
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();
}
```

O método `processXMLData()` é chamado quando o arquivo inteiro é lido no buffer de leitura (quando o objeto FileStream despacha o evento `complete`). Ele chama o método `readUTFBytes()` para obter uma versão de sequência dos dados lidos e cria um objeto XML, `prefsXML`, com base nessa sequência.

Para visualizar um aplicativo de amostra que mostre esses recursos, consulte Leitura e gravação de um arquivo de preferências XML.

Exemplo: Leitura e gravação de dados com acesso aleatório

Adobe AIR 1.0 e posterior

Arquivos MP3 podem incluir marcas ID3, que são seções no início ou no final do arquivo, contendo metadados que identificam a gravação. O próprio formato de marca ID3 tem revisões diferentes. Este exemplo descreve como fazer a leitura e a gravação de um arquivo MP3 que contém o formato ID3 mais simples (ID3 versão 1.0) usando o "acesso aleatório a dados do arquivo", o que significa que ele lê a partir de e grava em locais arbitrários no arquivo.

O arquivo MP3 que contém uma marca ID3 versão 1 inclui dados ID3 no final do arquivo, nos últimos 128 bytes.

Ao acessar um arquivo para acesso de leitura e gravação aleatório, é importante especificar `FileMode.UPDATE` como o parâmetro `fileMode` do método `open()` ou `openAsync()`:

```
var file:File = File.documentsDirectory.resolvePath("My Music/Sample ID3 v1.mp3");
var fileStr:FileStream = new FileStream();
fileStr.open(file, FileMode.UPDATE);
```

Isso permite ler e gravar no arquivo.

Ao abrir o arquivo, você pode definir o ponteiro `position` para a posição de 128 bytes antes do final do arquivo:

```
fileStr.position = file.size - 128;
```

Esse código define a propriedade `position` para esse local no arquivo, porque o formato ID3 v1.0 especifica que os dados da marca ID3 são armazenados nos últimos 128 bytes do arquivo. A especificação também diz o seguinte:

- Os primeiros 3 bytes da marca contêm a sequência `"TAG"`.
- Os 30 caracteres seguintes contêm o título da trilha MP3, como uma sequência.

- Os 30 caracteres seguintes contêm o nome do artista, como uma sequência.
- Os 30 caracteres seguintes contêm o nome do álbum, como uma sequência.
- Os 4 caracteres seguintes contêm o ano, como uma sequência.
- Os 30 caracteres seguintes contêm o comentário, como uma sequência.
- O byte seguinte contém um código indicando o gênero da trilha.
- Todos os dados de texto estão no formato ISO 8859-1.

O método `id3TagRead()` verifica os dados após serem lidos (no evento `complete`):

```
 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);
    }
}
```

Você também pode realizar uma gravação com acesso aleatório no arquivo. Por exemplo, você pode analisar a variável `id3Title` para assegurar que ela esteja com as maiúsculas e minúsculas corretas (usando métodos da classe String) e, em seguida, fazer a gravação de uma sequência modificada, chamada `newTitle`, no arquivo, como no seguinte:

```
fileStr.position = file.length - 125;    // 128 - 3
fileStr.writeMultiByte(newTitle, "iso-8859-1");
```

Para corresponder à versão 1 padrão do ID3, o tamanho da sequência `newTitle` deve ter 30 caracteres, preenchidos no final com o caractere de código 0 (`String.fromCharCode(0)`).

Capítulo 39: Armazenamento de dados locais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Utilize a classe SharedObject para armazenar pequenas quantidades de dados no computador do cliente. No Adobe AIR, você também pode usar a classe EncryptedLocalStore para armazenar pequenas quantidades de dados de privacidade do usuário no computador local em um aplicativo do AIR.

Você também pode ler e gravar arquivos no sistema de arquivos e (no Adobe AIR) acessar os arquivos locais do banco de dados. Para obter mais informações, consulte "Trabalho com o sistema de arquivos" na página 673 e "Trabalho com bancos de dados SQL locais no AIR" na página 737.

Há uma série de fatores de segurança que se relacionam com os objetos compartilhados. Para obter mais informações, consulte "Objetos compartilhados" na página 1114 em "Segurança" na página 1079.

Objetos compartilhados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um objeto compartilhado, às vezes referenciado como um "cookie Flash". é um arquivo de dados que pode ser criado no computador pelos sites visitados. Objetos compartilhados são usados com maior frequência para aprimorar sua experiência de navegação na Web, por exemplo, permitindo personalizar a aparência de um site visitado com frequência.

Sobre os objetos compartilhados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os objetos compartilhados funcionam como os cookies do navegador. Utilize a classe SharedObject para armazenar dados no disco rígido local do usuário e chamar dados durante a mesma sessão ou em uma sessão posterior. Os aplicativos podem acessar apenas seus próprios dados SharedObject, e somente quando executam no mesmo domínio. Os dados não foram enviados ao servidor e não estão acessíveis por outros aplicativos em execução nos outros domínios, mas podem tornar-se acessíveis por aplicativos do mesmo domínio.

Objetos compartilhados com cookies

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Cookies e objetos compartilhados são muito similares. Como a maioria dos programadores da Web estão familiarizados com o modo como os cookies funcionam, pode ser útil fazer uma analogia entre os cookies e os SharedObjects locais.

Os cookies que aderem ao padrão RFC 2109 geralmente têm as seguintes propriedades:

- Podem expirar e o fazem com frequência, por padrão, ao final da sessão.
- Podem ser desativados pelo cliente em uma base específica por site.

- Há um limite total de 300 cookies e 20 é o máximo por site.
- Normalmente, estão limitados ao tamanho de 4 KB cada.
- Algumas vezes, são percebidos como uma ameaça à segurança e, em resultado disso, ficam desativados no cliente.
- São armazenados em um local especificado pelo navegador cliente.
- São transmitidos do cliente ao servidor via HTTP.

Por outro lado, os objetos compartilhados apresentam as seguintes propriedades:

- Não expiram por padrão.
- Por padrão, estão limitados ao tamanho de 100 KB cada.
- Podem armazenar tipos de dados simples (como sequência de caracteres, matriz e data).
- São armazenados em um local especificado pelo aplicativo (dentro do diretório base do usuário).
- Nunca são transmitidos entre o cliente e o servidor.

Sobre a classe SharedObject

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Utilize a classe SharedObject para criar e excluir objetos compartilhados e também, detectar o tamanho atual do objeto SharedObject que está sendo utilizado.

Criação de um objeto compartilhado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para criar um objeto SharedObject utilize o método `SharedObject.getLocal()`, que possui a sintaxe a seguir:

```
SharedObject.getLocal("objectName" [, pathname]): SharedObject
```

O exemplo a seguir cria um objeto compartilhado denominado mySO:

```
public var mySO:SharedObject;
mySO = SharedObject.getLocal("preferences");
```

Isso cria um arquivo na máquina do cliente denominado preferences.sol.

O termo *local* faz referência ao local do objeto compartilhado. Nesse caso, o Adobe® Flash® Player armazena o arquivo SharedObject localmente no diretório base do cliente.

Quando você cria um objeto compartilhado, o Flash Player cria um novo diretório para o aplicativo e o domínio dentro da sua caixa de proteção. Cria, também, um arquivo *.sol que armazena os dados do SharedObject. O local padrão desse arquivo é um subdiretório no diretório base do usuário. A tabela a seguir apresenta os locais padrão desse diretório:

Sistema operacional	Localização
Windows 95/98/ME/2000/XP	`c:/Documents and Settings/`*`username`*`/Application Data/Macromedia/Flash Player/#SharedObjects`

Sistema operacional	Localização
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`

Abaixo do diretório #SharedObjects, está um diretório nomeado randomicamente. Abaixo desse diretório, vem um diretório que corresponde ao nome do host, em seguida, o caminho para o aplicativo e, finalmente, o arquivo *.sol.

Por exemplo, se você solicitar um aplicativo denominado MyApp.swf no host local, dentro de um subdiretório denominado /sos, o Flash Player armazenará o arquivo *.sol no seguinte local (no Windows XP):

```
c:/Documents and Settings/fred/Application Data/Macromedia/Flash
Player/#SharedObjects/KROKWXRK/#localhost/sos/MyApp.swf/data.sol
```

Nota: *Se você não prover um nome no método* `SharedObject.getLocal()`*, o Flash Player nomeará o arquivo .sol indefinido.*

Por padrão, o Flash pode salvar objetos SharedObject persistentes de até 100 KB por domínio. Esse valor pode ser configurado pelo usuário. Quando o aplicativo tenta salvar dados para um objeto compartilhado que ultrapassariam 100 KB, o Flash Player exibe a caixa de diálogo Armazenamento local, oferecendo a opção ao usuário de permitir ou negar o armazenamento local para o domínio que está solicitando o acesso.

Especificação de um arquivo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode utilizar o parâmetro opcional *pathname* para especificar um local no arquivo SharedObject. Esse arquivo deve estar em um subdiretório do diretório SharedObject desse domínio. Por exemplo, se você solicitar um aplicativo no host local e especificar:

```
mySO = SharedObject.getLocal("myObjectFile","/");
```

O Flash Player gravará o arquivo SharedObject no diretório /#localhost (ou /localhost, se o aplicativo estiver offline). Isso será útil caso você deseje que mais de um aplicativo no cliente seja capaz de acessar o mesmo objeto compartilhado. Nesse caso, o cliente pode executar dois aplicativos Flex, que especifiquem um caminho para o objeto compartilhado que esteja na raiz do domínio, e acessar o mesmo objeto compartilhado em ambos os aplicativos. Para compartilhar dados entre mais de um aplicativo sem persistência, é possível usar o objeto LocalConnection.

Se você especificar um diretório não existente, o Flash Player não criará um arquivo SharedObject.

Adição de dados a um objeto compartilhado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Adicione dados a um arquivo SharedObject *.sol utilizando a propriedade `data` do objeto SharedObject. Para adicionar novos dados ao objeto compartilhado, use a seguinte sintaxe:

```
sharedObject_name.data.variable = value;
```

O exemplo a seguir inclui as propriedades `userName`, `itemNumbers` e `adminPrivileges` e seus valores, em um SharedObject:

```
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;
```

Depois de designar valores à propriedade `data`, instrua o Flash Player a gravá-los no arquivo do SharedObject. Para forçar o Flash Player a gravar os valores no arquivo SharedObject, use o método `SharedObject.flush()`, como segue:

```
mySO.flush();
```

Se você não chamar o método `SharedObject.flush()`, o Flash Player gravará os valores no arquivo quando o aplicativo for encerrado. No entanto, isso não concederá ao usuário a oportunidade de aumentar o espaço disponível que o Flash Player terá para armazenar os dados, caso estes excedam as configurações padrão. Assim, chamar `SharedObject.flush()` é uma boa pedida.

Ao usar o método `flush()` para gravar objetos compartilhados em uma unidade de disco rígido do usuário, verifique atentamente se o usuário desativou explicitamente o armazenamento local usando o Gerenciador de configurações do Flash Player (www.macromedia.com/support/documentation/br/flashplayer/help/settings_manager07.html), conforme mostrado no exemplo a seguir:

```
var so:SharedObject = SharedObject.getLocal("test");
trace("Current SharedObject size is " + so.size + " bytes.");
so.flush();
```

Armazenamento de objetos em objetos compartilhados

Você pode armazenar objetos simples, como Matrizes ou Sequências de caracteres, em uma propriedade `data` do SharedObject.

O exemplo a seguir é uma classe ActionScript que define os métodos que controlam a interação com o objeto compartilhado. Estes métodos permitem que o usuário adicione e remova objetos do objeto compartilhado. Esta classe armazena uma ArrayCollection que contém objetos simples.

```
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();
        }
    }

}
```

O aplicativo Flex a seguir cria uma instância da classe ActionScript para cada um dos tipos de objetos compartilhados de que necessita. Em seguida, chama métodos nessa classe quando o usuário adiciona ou remove blogs ou URLs de sites.

```
<?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>
```

Armazenamento de objetos com tipo em objetos compartilhados.

É possível armazenar instâncias do ActionScript com tipo em objetos compartilhados. Isso é feito chamando o método `flash.net.registerClassAlias()` para registrar a classe. Se você criar uma instância da classe e armazená-la no membro de dados do objeto compartilhado e, posteriormente, ler o objeto, obterá uma instância com tipo. Por padrão, a propriedade `objectEncoding` do SharedObject tem suporte para a codificação AMF3 e extrai a instância armazenada do objeto SharedObject; a instância armazenada retém o mesmo tipo especificado quando você chamou o método `registerClassAlias()`.

(Somente iOS) Impede o backup em nuvem de objetos compartilhados locais

Adobe AIR 3.7 e posterior, somente iOS

Você pode definir a propriedade `SharedObject.preventBackup` para controlar se os objetos compartilhados locais serão armazenados no serviço de backup em nuvem do iOS. Trata-se de uma exigência da Apple para conteúdos que possam ser restaurados ou novamente baixados, mas que constituem um requisito para o funcionamento adequado de seu aplicativo durante o uso offline.

Criação de vários objetos compartilhados

É possível criar vários objetos compartilhados para o mesmo aplicativo Flex. Para fazer isso, designe a cada um deles um nome da instância diferente, como no exemplo a seguir:

```
public var mySO:SharedObject = SharedObject.getLocal("preferences");
public var mySO2:SharedObject = SharedObject.getLocal("history");
```

Isso criará os arquivos preferences.sol e history.sol no diretório local do aplicativo Flex.

Criação de um SharedObject seguro

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao criar um SharedObject remoto ou local usando `getLocal()` ou `getRemote()`, há um parâmetro opcional denominado `secure` que determina se o acesso a esse objeto compartilhado é restrito para arquivos SWF que são entregues por uma conexão HTTPS. Se esse parâmetro estiver definido como `true` e o arquivo SWF for entregue por HTTPS, o Flash Player criará um novo objeto compartilhado protegido ou obterá uma referência a um objeto compartilhado protegido existente. Esse objeto compartilhado protegido pode ser lido ou gravado apenas por arquivos SWF entregues por HTTPS que chamam `SharedObject.getLocal()` com o parâmetro secure definido como `true`. Se esse parâmetro estiver definido como `false` e o arquivo SWF for entregue por HTTPS, o Flash Player criará um novo objeto compartilhado ou obterá uma referência a um objeto compartilhado existente.

Esse objeto compartilhado pode ser lido ou gravado por arquivos SWF entregues por conexões não HTTPS. Se o arquivo SWF for entregue por uma conexão não HTTPS e você tentar definir esse parâmetro como `true`, haverá uma falha na criação de um novo objeto compartilhado (ou o acesso de um objeto compartilhado protegido criado anteriormente), um erro será emitido e o objeto compartilhado será definido como `null`. Se você tentar executar o seguinte snippet em uma conexão não HTTPS, o método `SharedObject.getLocal()` emitirá um erro:

```
try
{
    var so:SharedObject = SharedObject.getLocal("contactManager", null, true);
}
catch (error:Error)
{
    trace("Unable to create SharedObject.");
}
```

Independentemente do valor desse parâmetro, os objetos compartilhados criados serão levados em consideração na contagem da quantidade total de espaço em disco permitida para um domínio.

Exibição de conteúdo de um objeto compartilhado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os valores são armazenados em objetos compartilhados dentro da propriedade `data`. Você pode executar um loop em cada valor dentro de uma ocorrência do objeto compartilhado usando um loop `for..in`, conforme mostrado no exemplo a seguir:

```
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]);
}
```

Destruição de objetos compartilhados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para destruir um SharedObject no cliente, use o método `SharedObject.clear()`. Isso não destruirá os diretórios no caminho padrão dos objetos compartilhados do aplicativo.

O exemplo a seguir exclui o arquivo SharedObject do cliente:

```
public function destroySharedObject():void {
    mySO.clear();
}
```

Exemplo de SharedObject

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo a seguir demonstra que é possível armazenar objetos simples como, por exemplo, Date em um objeto SharedObject sem a necessidade de serializar e desserializar esses objetos.

Este exemplo inicia dando boas-vindas a você pela sua primeira visita. Assim que você clicar em Log Out, o aplicativo armazenará a data atual em um objeto compartilhado. Na próxima vez que iniciar o aplicativo ou atualizar a página, o aplicativo dará boas-vindas pelo seu retorno, com um lembrete sobre o tempo em que esteve desconectado.

Para ver o aplicativo em ação, inicie o aplicativo, clique em Log Out e atualize a página. O aplicativo exibirá a data e hora em que você clicou no botão Log Out na visita anterior. A qualquer momento, é possível excluir as informações armazenadas clicando no botão Excluir 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>
```

Armazenamento local criptografado

A classe EncryptedLocalStore (ELS) oferece um mecanismo de armazenamento criptografado que você pode usar como um pequeno cache para os dados privados de um aplicativo. Os dados de ELS não podem ser compartilhados entre aplicativos. O objetivo do ELS é permitir que um aplicativo armazene com facilidade itens criados, como credenciais de login e outras informações privadas. Os dados de ELS não devem ser considerados permanentes, conforme é descrito em "Limitações do armazenamento local criptografado" e em "Melhores práticas" abaixo.

Nota: *Além do armazenamento local criptografado, o AIR também fornece criptografia para conteúdo armazenado em bancos de dados do SQL. Para obter detalhes, consulte* *"Uso da criptografia com bancos de dados SQL" na página 783.*

Você pode querer utilizar o armazenamento local criptografado para colocar em cache informações que precisam ser protegidas como, por exemplo, credenciais de login para serviços web. O ELS é apropriado para armazenar informações que precisam ser mantidas e forma particular. No entanto, ele não protege os dados contra outros processos executados da mesma conta de usuário. Por esse motivo, ele não é apropriado para proteger dados secretos de aplicativos como, por exemplo, informações de DRM e chaves de criptografia.

Em plataformas de computação pessoal, o AIR usa DPAPI no Windows, KeyChain no Mac OS e no iOS e KeyRing ou KWallet no Linux para associar o armazenamento local criptografado a cada aplicativo e usuário. O armazenamento local criptografado usa a criptografia AES-CBC de 128 bits.

No Android, os dados armazenados pela classe EncryptedLocalStorage não são criptografados. Em vez disso, os dados são protegidos pela segurança em nível de usuário fornecida pelo sistema operacional. O sistema operacional Android atribui um ID de usuário separado para cada aplicativo. Os aplicativos podem acessar somente seus próprios arquivos e arquivos criados em locais públicos (tais como cartões de armazenamento removíveis). Observe que nos dispositivos raiz do Android, os aplicativos que operam com privilégios de raiz PODEM acessar os arquivos de outros aplicativos. Dessa forma, em um dispositivo enraizado, o armazenamento local criptografado não fornece um nível muito alto de proteção de dados como ocorre em um dispositivo não-enraizado.

As informações no armazenamento local criptografado estão disponíveis apenas a conteúdo de aplicativos AIR na caixa de proteção de segurança do aplicativo.

Se você atualizar um aplicativo do AIR, a versão atualizada mantém o acesso dos dados existentes no armazenamento local criptografado, a não ser que:

- Itens adicionados com o parâmetro `stronglyBound` definido em `true`
- As versões existentes e atualizadas são publicadas com antecedência no AIR 1.5.3 e a atualização é assinada com uma assinatura de migração.

Limitações do armazenamento local criptografado

Os dados no armazenamento local criptografado são protegidos pelas credenciais da conta de sistema operacional do usuário. Outras entidades não podem acessar os dados no armazenamento a menos que possam fazer login como aquele usuário. No entanto, os dados não ficam protegidos contra o acesso de outros aplicativos executados por um usuário autenticado.

Como o usuário deve ser autenticado para que esses ataques funcionem, os dados privados do usuário ainda estão protegidos (a menos que a própria conta do usuário esteja comprometida). No entanto, os dados que o seu aplicativo deseja manter em segredo, como as chaves usadas para licenciamento ou o gerenciamento de direitos digitais, não estão seguros. Assim, o ELS não é um local adequado para armazenar essas informações. Isto é somente um local apropriado para armazenar dados particulares do usuário como, por exemplo, senhas.

Os dados no ELS podem ser perdidos por diversos motivos. Por exemplo, o usuário pode desinstalar o aplicativo e excluir o arquivo criptografado. Ou o ID do publicador pode ser alterado como resultado de uma atualização. Dessa forma, o ELS deve ser tratado como um cache particular e não um armazenamento de dados permanente.

O parâmetro `stronglyBound` foi descontinuado e não deve ser definido como `true`. Nenhuma proteção adicional é fornecida aos dados se o parâmetro for definido como `true`. Ao mesmo tempo, o acesso aos dados é perdido sempre que um aplicativo é atualizado — mesmo se o ID do publicador permanecer o mesmo.

O armazenamento local criptografado poderá ficar mais lento se os dados armazenados excederem 10 MB.

Quando você desinstala um aplicativo do AIR, o desinstalador não exclui os dados armazenados no armazenamento local criptografado.

Melhores práticas

Entre as melhores práticas para usar o ELS estão:

- Use o ELS para armazenar dados importantes do usuário, como senhas (definindo stronglyBound como false)
- Não use o ELS para armazenar os segredos dos aplicativos como chaves de DRM ou tokens de licenciamento.
- Fornece uma forma de seu aplicativo recriar os dados armazenados no ELS se os dados do ELS forem perdidos. Por exemplo, ao solicitar que o usuário digite novamente as credenciais de conta quando necessário.
- Não utilizar o parâmetro `stronglyBound`.
- Se você definir o parâmetro `stronglyBound` como `true`, não migre os itens armazenados durante uma atualização. Em vez disso, recrie os dados após a atualização.
- Somente armazene quantidades relativamente pequenas de dados. Para grandes quantidades de dados, utilize o banco de dados SQL do AIR criptografia.

Mais tópicos da Ajuda

flash.data.EncryptedLocalStore

Inclusão de dados no armazenamento local criptografado

Use o método estático `setItem()` da classe EncryptedLocalStore para armazenar os dados no armazenamento local. Os dados são armazenados em uma tabela de hash, usando sequências de caracteres como chaves, com os dados armazenados como matrizes de bytes.

Por exemplo, o código a seguir armazena uma sequência de caracteres no armazenamento local criptografado:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
```

O terceiro parâmetro do método `setItem()`, o parâmetro `stronglyBound`, é opcional. Quando este parâmetro está definido em `true`, o armazenamento local criptografado associa o item armazenado ao aplicativo aos bits e à assinatura do aplicativo do AIR:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes, false);
```

Para um item armazenado com `stronglyBound` definido como `true`, as chamadas subsequentes a `getItem()` apenas são bem-sucedidas se o aplicativo do AIR de chamada for idêntico ao aplicativo de armazenamento (se nenhum dado nos arquivos do diretório do aplicativo tiver sido alterado). Se o aplicativo do AIR de chamada for diferente do aplicativo de armazenamento, o aplicativo lançará uma exceção de erro quando você chamar `getItem()` para um item fortemente ligado. Se você atualizar seu aplicativo, ele não será capaz de ler os dados fortemente ligados previamente gravados no armazenamento local criptografado. A configuração de `stronglyBound` como `true` em dispositivos móveis é ignorada; o parâmetro sempre é tratado como `false`.

Se o parâmetro `stronglyBound` estiver definido como `false` (o padrão), só o ID da editora precisa se manter igual para que o aplicativo leia os dados. Os bits do aplicativo podem mudar (e precisam ser atribuídos pelo mesmo editor), mas não precisam ser exatamente os mesmos bits que eram no aplicativo que armazenou os dados. Aplicativos atualizados com o mesmo ID do publicado do original podem continuar a acessar os dados.

Nota: *Na prática, configurar* `stronglyBound` *para* `true` *não adiciona nenhuma proteção adicional aos dados. Um usuário mal intencionado ainda pode alterar um aplicativo para obter acesso a itens gravados no ELS. Além disso, os dados são protegidos de ameaças externas, mesmo se* `stronglyBound` *estiver definido em* `true` *ou* `false`*. Por esses motivos, não encorajamos definir* `stronglyBound` *em* `true`*.*

Acesso aos dados no armazenamento local criptografado

Adobe AIR 1.0 e posterior

Você pode recuperar um valor do armazenamento local criptografado usando o método `EncryptedLocalStore.getItem()`, como no seguinte exemplo:

```
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length)); // "Bob"
```

Remoção de dados do armazenamento local criptografado

Adobe AIR 1.0 e posterior

Você pode excluir um valor do armazenamento local criptografado usando o método `EncryptedLocalStore.removeItem()`, como no seguinte exemplo:

```
EncryptedLocalStore.removeItem("firstName");
```

Você pode limpar todos os dados do armazenamento local criptografado chamando o método `EncryptedLocalStore.reset()`, como no seguinte exemplo:

```
EncryptedLocalStore.reset();
```

Capítulo 40: Trabalho com bancos de dados SQL locais no AIR

Adobe AIR 1.0 e posterior

O Adobe® AIR® permite criar e trabalhar com bancos de dados SQL locais. O tempo de execução inclui um mecanismo de banco de dados SQL com suporte para muitos recursos SQL padrão através do sistema de banco de dados de código-fonte aberto SQLite. Um banco de dados SQL local pode ser usado para armazenar dados locais persistentes. Por exemplo, ele pode ser usado para dados de aplicativo, configurações de usuários de aplicativo, documentos ou qualquer outro tipo de dados que você desejar que o aplicativo salve localmente.

Sobre bancos de dados SQL locais

Adobe AIR 1.0 e posterior

Veja uma explicação rápida e exemplos de códigos para o uso de bancos de dados SQL nos seguintes artigos de apresentação rápida do Adobe Developer Connection:

- Trabalho assíncrono com um banco de dados SQL local (Flex)
- Trabalhar sincronamente com um banco de dados SQL local) (Flex)
- Usar banco de dados criptografado (Flex)
- Trabalhar assincronamente com um banco de dados SQL local (Flash)
- Trabalhar sincronamente com um banco de dados SQL local (Flash)
- Usar banco de dados criptografado (Flash)

O Adobe AIR inclui um mecanismo de banco de dados relacional baseado em SQL que é executado no tempo de execução, com dados armazenados localmente em arquivos de banco de dados no computador em que o aplicativo do AIR é executado (no disco rígido do computador, por exemplo). Como a execução do banco de dados ocorre localmente, da mesma forma que o armazenamento dos arquivos de dados, um banco de dados pode ser usado por um aplicativo do AIR independentemente de haver uma conexão de rede disponível. Por isso, o mecanismo de banco de dados SQL local do tempo de execução é conveniente para armazenar dados de aplicativo locais persistentes, principalmente se você tem experiência em SQL e bancos de dados relacionais.

Usos de bancos de dados SQL locais

Adobe AIR 1.0 e posterior

A funcionalidade de banco de dados SQL local do AIR pode ser utilizada para qualquer finalidade quando você quiser armazenar dados de aplicativo no computador local de um usuário. O Adobe AIR inclui diversos mecanismos para armazenar dados localmente, e cada um deles tem suas próprias vantagens. Veja abaixo alguns dos usos possíveis de um banco de dados SQL local no aplicativo do AIR:

- No caso de um aplicativo orientado a dados (um catálogo de endereços, por exemplo), é possível usar um banco de dados para armazenar os principais dados de aplicativo.

- No caso de um aplicativo orientado a documento, em que os usuários criam documentos para salvá-los e possivelmente compartilhá-los, cada documento pode ser salvo como um arquivo de banco de dados em um local designado pelo usuário. (Observe, no entanto, que a menos que o banco de dados seja criptografado, qualquer aplicativo do AIR poderia abrir o arquivo do banco de dados. A criptografia é recomendada para documentos potencialmente confidenciais.)
- No caso de um aplicativo com reconhecimento de rede, é possível usar um banco de dados para armazenar um cache local de dados de aplicativo ou para armazenar dados temporariamente quando não há uma conexão de rede disponível. Você pode criar um mecanismo para sincronizar o banco de dados local com o armazenamento de dados da rede.
- No caso de qualquer aplicativo, um banco de dados pode ser usado para armazenar configurações de aplicativo de um usuário, como opções ou informações do aplicativo (por exemplo, o tamanho e a posição das janelas).

Mais tópicos da Ajuda

Christophe Coenraets: diretório de funcionários no AIR for Android

Raymond Camden: jQuery e AIR - Movimentação da página da Web para o aplicativo

Sobre bancos de dados do AIR e arquivos de banco de dados

Adobe AIR 1.0 e posterior

Um banco de dados SQL local do Adobe AIR é armazenado como um único arquivo no sistema de arquivos do computador. O tempo de execução inclui o mecanismo de banco de dados SQL que gerencia a criação e a estruturação de arquivos de banco de dados, bem como a manipulação e a recuperação de dados de um arquivo de banco de dados. O tempo de execução não especifica como ou onde os dados de banco de dados são armazenados no sistema de arquivos; cada banco de dados é armazenado por completo em um único arquivo. Você especifica o local no sistema de arquivos em que o arquivo do banco de dados deve ser armazenado. Um único aplicativo do AIR pode acessar um ou vários bancos de dados separados (isto é, arquivos de banco de dados separados). Como o tempo de execução armazena cada banco de dados como um único arquivo no sistema de arquivos, você pode localizar o seu banco de dados quando necessário pelo design das restrições do sistema operacional quanto ao acesso a aplicativos e arquivos. Cada usuário pode ter um arquivo de banco de dados à parte para seus dados específicos, ou um arquivo de banco de dados pode ser acessado por todos os usuários do aplicativo em um único computador para dados compartilhados. Como os dados estão armazenados localmente em um único computador, eles não são compartilhados automaticamente em diferentes computadores. O mecanismo de banco de dados SQL local não tem um recurso que permita executar instruções SQL em um banco de dados remoto ou baseado em servidor.

Sobre bancos de dados relacionais

Adobe AIR 1.0 e posterior

Um banco de dados relacional consiste em um mecanismo para armazenar (e recuperar) dados em um computador. Os dados são organizados em tabelas: as linhas representam registros ou itens, e as colunas (também chamadas de “campos”) dividem cada registro em valores individuais. Por exemplo, um aplicativo de catálogo de endereços pode conter uma tabela chamada “amigos”. Cada linha da tabela representa um único amigo armazenado no banco de dados. As colunas da tabela representam dados como nome, sobrenome, data de nascimento e assim por diante. Para cada linha de amigo na tabela, o banco de dados armazena um valor em separado para cada coluna.

Os bancos de dados relacionais têm a finalidade de armazenar dados complexos, em que um item é associado ou relacionado a itens de outro tipo. Em um banco de dados relacional, qualquer dado que tem um relacionamento um para muitos — em que um único registro pode estar relacionado a vários registros de um tipo diferente — deve ser dividido entre diferentes tabelas. Por exemplo, suponha que você queira que o aplicativo de catálogo de endereços armazene vários números de telefone de cada amigo; isto é um relacionamento um para muitos. A tabela “amigos” contém todas as informações pessoais de cada amigo. Uma tabela à parte chamada “números de telefone” contém todos os números de telefone de todos os amigos.

Além de armazenar os dados de amigos e os números de telefone, cada tabela precisa de dados para controlar os relacionamentos entre as duas tabelas — a fim de comparar registros de amigo individuais com seus números de telefone. Esses dados são chamados de chave primária, um identificador exclusivo que diferencia cada linha de uma tabela das demais linhas dessa tabela. A chave primária pode ser uma “chave natural”, ou seja, um dos itens de dados que diferencia cada registro de uma tabela naturalmente. Na tabela “amigos”, se você souber que nenhum dos seus amigos tem a mesma data de aniversário, poderá usar a coluna de data de nascimento como a chave primária (uma chave natural) dessa tabela. Se não houver uma chave natural, crie uma coluna de chave primária em separado, como “id de amigo”: um valor artificial que o aplicativo usa para diferenciar entre as linhas.

Utilizando uma chave primária, você pode configurar relacionamentos entre várias tabelas. Por exemplo, suponha que na tabela "amigos" exista uma coluna chamada "id de amigo", que contém um número exclusivo para cada linha (cada amigo). A tabela “números de telefone” relacionada pode ser estruturada com duas colunas: uma com o “id de amigo” do amigo a quem pertence o número de telefone e outra com o número de telefone propriamente dito. Assim, independentemente de quantos números de telefones um só amigo tiver, todos poderão ser armazenados na tabela “números de telefone” e vinculados ao amigo relacionado através da chave primária “id de amigo”. Quando uma chave primária de uma tabela é usada em uma tabela relacionada para especificar a conexão entre os registros, o valor contido na tabela é chamado de chave externa. Diferentemente de muitos bancos de dados, o mecanismo de banco de dados local do AIR não permite criar restrições de chave externa, que são restrições que automaticamente verificam se um valor de chave externa inserido ou atualizado tem uma linha correspondente na tabela de chave primária. Todavia, os relacionamentos de chave externa são uma parte importante da estrutura de um banco de dados relacional, e as chaves externas devem ser usadas quando você cria relacionamentos entre as tabelas do banco de dados.

Sobre a SQL

Adobe AIR 1.0 e posterior

A SQL (linguagem de consulta estruturada) é usada com bancos de dados relacionais para manipular e recuperar dados. A SQL é uma linguagem descritiva e não de procedimento. Em vez de dar ao computador instruções sobre como recuperar dados, uma instrução SQL descreve o conjunto de dados desejado. O mecanismo de banco de dados determina como recuperar esses dados.

A linguagem SQL foi padronizada pelo ANSI (American National Standards Institute). O banco de dados SQL local do Adobe AIR oferece suporte à maior parte do padrão SQL-92.

Para obter descrições específicas da linguagem SQL suportada no Adobe AIR, consulte “Suporte SQL em bancos de dados locais” na página 1148.

Sobre classes de banco de dados SQL

Adobe AIR 1.0 e posterior

Para trabalhar com bancos de dados SQL locais no ActionScript 3.0, você deve usar ocorrências destas classes do pacote flash.data:

Classe	Descrição
flash.data.SQLConnection	Oferece um modo de criar e abrir bancos de dados (arquivos de banco de dados), bem como métodos para executar operações no nível de banco de dados e controlar transações de banco de dados.
flash.data.SQLStatement	Representa uma única instrução SQL (uma única consulta ou comando) que é executada em um banco de dados, incluindo a definição do texto da instrução e a configuração dos valores de parâmetro.
flash.data.SQLResult	Oferece um modo de obter informações sobre execução de uma instrução ou os resultados dela, como as linhas de resultado de uma instrução `SELECT`, o número de linhas afetadas por uma instrução `UPDATE` ou `DELETE` e assim por diante.

Para obter informações de esquema que descrevam a estrutura de um banco de dados, use estas classes do pacote flash.data:

Classe	Descrição
flash.data.SQLSchemaResult	Funciona como um contêiner de resultados do esquema de banco de dados gerados pela chamada do método `SQLConnection.loadSchema()`.
flash.data.SQLTableSchema	Fornece informações que descrevem uma única tabela de um banco de dados.
flash.data.SQLViewSchema	Fornece informações que descrevem uma única visualização de um banco de dados.
flash.data.SQLIndexSchema	Fornece informações que descrevem uma única coluna de uma tabela ou visualização de um banco de dados.
flash.data.SQLTriggerSchema	Fornece informações que descrevem um único disparador de um banco de dados.

Outras classes do pacote flash.data fornecem constantes que são utilizadas com as classes SQLConnection e SQLColumnSchema:

Classe	Descrição
flash.data.SQLMode	Define um conjunto de constantes que representam os possíveis valores do parâmetro `openMode` dos métodos `SQLConnection.open()` e `SQLConnection.openAsync()`.
flash.data.SQLColumnNameStyle	Define um conjunto de constantes que representam os possíveis valores da propriedade `SQLConnection.columnNameStyle`.
flash.data.SQLTransactionLockType	Define um conjunto de constantes que representam os possíveis valores do parâmetro de opção do método `SQLConnection.begin()`.
flash.data.SQLCollationType	Define um conjunto de constantes que representam os valores possíveis para a propriedade `SQLColumnSchema.defaultCollationType` e o parâmetro `defaultCollationType` do construtor `SQLColumnSchema()`.

Além disso, as seguintes classes do pacote flash.events representam os eventos (e as constantes de suporte) que você utiliza:

Classe	Descrição
flash.events.SQLEvent	Define os eventos que uma ocorrência de SQLConnection ou SQLStatement despacha quando alguma de suas operações é executada com êxito. Cada operação tem uma constante de tipo de evento associada definida na classe SQLEvent.
flash.events.SQLErrorEvent	Define o evento que uma ocorrência de SQLConnection ou SQLStatement despacha quando alguma de suas operações resulta em erro.
flash.events.SQLUpdateEvent	Define o evento que uma ocorrência de SQLConnection despacha quando dados de tabela de um dos bancos de dados conectados são alterados como resultado da execução de uma instrução SQL `INSERT`, `UPDATE` ou `DELETE`.

Para finalizar, as seguintes classes do pacote flash.errors fornecem informações sobre erros de operação do banco de dados:

Classe	Descrição
flash.errors.SQLError	Fornece informações sobre um erro de funcionamento do banco de dados, incluindo a operação que foi tentada e a causa da falha.
flash.errors.SQLErrorOperation	Define um conjunto de constantes que representam os valores possíveis para a propriedade `operation` da classe SQLError, que indica a operação de banco de dados que resultou em erro.

Sobre modos de execução síncrona e assíncrona

Adobe AIR 1.0 e posterior

Quando você cria código para trabalhar com um banco de dados SQL local, especifica que a execução das operações de banco de dados ocorra de um destes dois modos de execução: modo de execução assíncrona ou síncrona. Em geral, os exemplos de código mostram como executar cada operação das duas maneiras, para que você possa usar o exemplo mais adequado às suas necessidades.

No modo de execução assíncrona, você dá uma instrução ao tempo de execução, que despacha um evento quando a operação solicitada é concluída ou quando falha. Primeiro você instrui o mecanismo de banco de dados a executar uma operação. O mecanismo de banco de dados trabalha em segundo plano enquanto o aplicativo continua a executar. Por último, quando a operação é concluída (ou se falha), o mecanismo de banco de dados despacha um evento. O seu código, disparado pelo evento, executa operações subsequentes. Esta abordagem tem uma vantagem significativa: o tempo de execução realiza as operações de banco de dados em segundo plano enquanto o código do aplicativo principal continua en execução. Se a operação de banco de dados demorar muito tempo, o aplicativo continuará executando. O mais importante é que o usuário pode continuar a interagir com ele sem que a tela congele. Entretanto, o código de operação assíncrona pode ser mais complexo de criar do que outro código. Essa complexidade geralmente ocorre quando várias operações dependentes devem ser divididas entre diversos métodos de ouvinte de evento.

Em termos conceituais, é mais fácil codificar operações como uma sequência simples de etapas (um conjunto de operações síncronas) e não como um conjunto de operações divididas em vários métodos de ouvinte de evento. Além das operações de banco de dados assíncronas, o Adobe AIR também permite executar operações de banco de dados de maneira síncrona. No modo de execução síncrona, as operações não são executadas em segundo plano. Elas ocorrem na mesma sequência de execução que todos os outros códigos de aplicativo. Você instrui o mecanismo de banco de dados a executar uma operação. Em seguida, o código pausa nesse ponto enquanto o mecanismo de banco de dados faz o seu trabalho. Quando a operação é concluída, a execução prossegue com a próxima linha do código que você criou.

O fato de as operações serem executadas de maneira assíncrona ou síncrona é definido no nível de SQLConnection. Usando uma única conexão de banco de dados, você não consegue executar algumas operações ou instruções em modo síncrono e outras em modo assíncrono. Você especifica se uma classe SQLConnection opera no modo de execução síncrono ou assíncrono chamando um método SQLConnection para abrir o banco de dados. Se você chamar `SQLConnection.open()`, a conexão funcionará no modo de execução síncrona; se você chamar `SQLConnection.openAsync()`, ela funcionará no modo de execução assíncrona. Uma vez que uma ocorrência de SQLConnection é conectada a um banco de dados usando `open()` ou `openAsync()`, ela é fixada ao modo de execução síncrona ou assíncrona, a menos que você feche e reabra a conexão com o banco de dados.

Cada modo de execução tem suas vantagens. Apesar da semelhança entre a maioria dos aspectos de cada modo, existem algumas diferenças das quais você deve se lembrar quando trabalhar neles. Para obter mais informações sobre estes tópicos, e sugestões de como trabalhar em cada modo, consulte "Uso de operações de banco de dados síncronas e assíncronas" na página 777.

Criação e modificação de um banco de dados

Adobe AIR 1.0 e posterior

Para que o seu aplicativo adicione ou recupere dados, deve haver um banco de dados com tabelas definidas nele que possam ser acessadas pelo aplicativo. Descrevemos aqui as tarefas de criação de um banco de dados e da estrutura de dados de um banco de dados. Embora usadas com menos frequência do que a inserção e a recuperação de dados, estas tarefas são necessárias para a maioria dos aplicativos.

Mais tópicos da Ajuda

Lembre-se do Flex: Carregamento de um banco de dados existente do AIR

Criação de um banco de dados

Adobe AIR 1.0 e posterior

Para criar um arquivo de banco de dados, primeiro você deve criar uma instância do SQLConnection. Chame o método `open()` correspondente para abri-la no modo de execução síncrona ou o método `openAsync()` para abri-la no modo de execução assíncrona. Os métodos `open()` e `openAsync()` são usados para abrir uma conexão com um banco de dados. Se você passar uma ocorrência de File que se refira a um local de arquivo não existente para o parâmetro `reference` (o primeiro parâmetro), o método `open()` ou `openAsync()` criará um arquivo de banco de dados no local do arquivo e abrirá uma conexão com o banco de dados recém-criado.

Independentemente de você chamar o método `open()` ou `openAsync()` para criar um banco de dados, o nome do arquivo do banco de dados poderá ser qualquer nome de arquivo válido com qualquer extensão de nome de arquivo. Se você chamar o método `open()` ou `openAsync()` com `null` para o parâmetro `reference`, será criado um novo banco de dados na memória e não um arquivo de banco de dados no disco.

A listagem de código a seguir mostra o processo de criação de um arquivo de banco de dados (um novo banco de dados) usando o modo de execução assíncrona. Nesse caso, o arquivo de banco de dados é salvo no "Apontar para o diretório de armazenamento do aplicativo" na página 694 com o nome de arquivo "DBSample.db":

```
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>
```

Nota: *Embora a classe File permita apontar para um caminho nativo específico de arquivo, fazer isso pode levar a aplicativos que não funcionam em várias plataformas. Por exemplo, o caminho C:\Documents and Settings\joe\test.db só funciona no Windows. Por essas razões, é melhor usar as propriedades estáticas da classe File, como o método* `File.applicationStorageDirectory` *e* `resolvePath()` *(como mostrado no exemplo anterior). Para obter mais informações, consulte* *"Caminhos de objetos File" na página 690.*

Para executar operações de maneira síncrona, quando abrir uma conexão de banco de dados com a ocorrência de SQLConnection, chame o método `open()`. O seguinte exemplo mostra como criar e abrir uma ocorrência de SQLConnection que execute suas operações de maneira síncrona:

```
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>
```

Criação de tabelas de banco de dados

Adobe AIR 1.0 e posterior

Criar uma tabela em um banco de dados envolve executar uma instrução SQL nesse banco de dados usando o mesmo processo seguido para executar uma instrução SQL, como `SELECT`, `INSERT`, entre outras. Para criar uma tabela, use uma instrução `CREATE TABLE`, que inclui definições de colunas e restrições para a nova tabela. Para obter mais informações sobre como executar instruções SQL, consulte "Trabalhar com instruções SQL" na página 753.

O exemplo a seguir demonstra como criar uma tabela chamada "employees" em um arquivo de banco de dados existente usando o modo de execução assíncrona. Observe que este código presume que há uma instância de SQLConnection denominada `conn` já instanciada e conectada a um banco de dados.

```
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>
```

O exemplo a seguir demonstra como criar uma tabela chamada “employees” em um arquivo de banco de dados existente usando o modo de execução síncrona. Observe que este código presume que há uma instância de SQLConnection denominada `conn` já instanciada e conectada a um banco de dados.

```
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>
```

Manipulação de dados de um banco de dados SQL

Adobe AIR 1.0 e posterior

Existem algumas tarefas comuns que você executa ao trabalhar com bancos de dados SQL locais. Essas tarefas incluem conectar-se a um banco de dados, adicionar dados a tabelas e recuperar dados de tabelas em um banco de dados. Também há vários problemas dos quais você deve estar ciente quando executa essas tarefas, como trabalhar com tipos de dados e tratar de erros.

Além disso, há diversas tarefas de banco de dados que envolvem coisas com as quais você lidará menos frequentemente, mas que deverá fazer para poder executar essas tarefas mais comuns. Por exemplo, para poder se conectar a um banco de dados e recuperar dados de uma tabela, você terá de criar o banco de dados e a estrutura de tabelas nele. Essas tarefas de configuração inicial menos frequentes são discutidas em "Criação e modificação de um banco de dados" na página 742.

Você pode optar por executar operações de banco de dados assincronamente, o que significa que o mecanismo de banco de dados é executado em segundo plano e despacha um evento para notificar o usuário quando a operação é bem-sucedida ou quando falha. Também é possível executar estas operações de forma síncrona. Nesse caso, as operações de banco de dados são executadas uma após a outra, e o aplicativo inteiro (inclusive atualizações da tela) espera até que as operações sejam concluídas para poder executar outro código. Para obter mais informações sobre como trabalhar no modo de execução assíncrona ou síncrona, consulte "Uso de operações de banco de dados síncronas e assíncronas" na página 777.

Conexão com um banco de dados

Adobe AIR 1.0 e posterior

Para que você possa executar qualquer operação de banco de dados, primeiro abra uma conexão com o arquivo de banco de dados. Uma instância de SQLConnection é usada para representar uma conexão com um ou mais bancos de dados. O primeiro banco de dados conectado usando uma ocorrência de SQLConnection é chamado de banco de dados "principal". Esse banco de dados é conectado através do método `open()` (para o modo de execução síncrona) ou do método `openAsync()` (para o modo de execução assíncrona).

Se você abrir um banco de dados usando a operação `openAsync()` assíncrona, registre-se para o evento `open` da ocorrência de SQLConnection para saber quando a operação `openAsync()` for concluída. Registre-se para o evento `error` da ocorrência de SQLConnection para determinar se a operação falhou.

O exemplo a seguir mostra como abrir um arquivo de banco de dados existente para execução assíncrona. O arquivo de banco de dados chama-se "DBSample.db" e está localizado no "Apontar para o diretório de armazenamento do aplicativo" na página 694 do usuário.

```
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>
```

O exemplo a seguir mostra como abrir um arquivo de banco de dados existente para execução síncrona. O arquivo de banco de dados chama-se “DBSample.db” e está localizado no “Apontar para o diretório de armazenamento do aplicativo” na página 694 do usuário.

```
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>
```

Observe que, na chamada do método `openAsync()` no exemplo de operação assíncrona e na chamada do método `open()` no exemplo de operação síncrona, o segundo argumento é a constante `SQLMode.UPDATE`. Especificar `SQLMode.UPDATE` para o segundo parâmetro (`openMode`) faz com que o tempo de execução despache um erro se o arquivo especificado não existe. Caso você passe `SQLMode.CREATE` para o parâmetro `openMode` (ou caso deixe o parâmetro `openMode` como off), o tempo de execução tentará criar um arquivo de banco de dados se o arquivo especificado não existir. No entanto, se o arquivo existir, será aberto, o que é o mesmo que usar `SQLMode.Update`. Também é possível especificar `SQLMode.READ` para o parâmetro `openMode` para abrir um banco de dados existente em modo somente leitura. Nesse caso, os dados podem ser recuperados do banco de dados, mas não é possível adicionar, excluir ou alterar nenhum dado.

Trabalhar com instruções SQL

Adobe AIR 1.0 e posterior

Uma instrução SQL individual (consulta ou comando) é representada no tempo de execução como um objeto SQLStatement. Siga estas etapas para criar e executar uma instrução SQL:

Crie uma ocorrência de SQLStatement.
O objeto SQLStatement representa a instrução SQL no seu aplicativo.

```
var selectData:SQLStatement = new SQLStatement();
```

Especifique em que banco de dados a consulta é executada.
Para isso, defina a propriedade `sqlConnection` do objeto SQLStatement como a ocorrência de SQLConnection que está conectada ao banco de dados desejado.

```
// A SQLConnection named "conn" has been created previously
selectData.sqlConnection = conn;
```

Especifique a instrução SQL propriamente dita.
Crie o texto da instrução como String e o atribua à propriedade `text` da ocorrência de SQLStatement.

```
selectData.text = "SELECT col1, col2 FROM my_table WHERE col1 = :param1";
```

Defina funções para trabalhar com o resultado da operação de execução (somente no modo de execução assíncrona).
Use o método `addEventListener()` para registrar funções como ouvintes para os eventos `result` e `error` da ocorrência de SQLStatement.

```
// 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
}
```

Se preferir, você pode especificar métodos de ouvinte usando um objeto Responder. Nesse caso, crie a ocorrência de Responder e vincule os métodos de ouvinte a ela.

```
// 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
}
```

Se o texto da instrução incluir definições de parâmetro, atribua valores para esses parâmetros.
Para atribuir valores de parâmetro, use a propriedade de matriz associativa `parameters` da ocorrência de SQLStatement.

```
selectData.parameters[":param1"] = 25;
```

Execute a instrução SQL.
Chame o método `execute()` da ocorrência de SQLStatement.

```
// using synchronous execution mode
// or listener methods in asynchronous execution mode
selectData.execute();
```

Além disso, se você estiver usando um Responder em vez de ouvintes de evento no modo de execução assíncrona, passe a ocorrência de Responder para o método `execute()`.

```
// using a Responder in asynchronous execution mode
selectData.execute(-1, selectResponder);
```

Para obter exemplos específicos que demonstrem estas etapas, consulte os seguintes tópicos:

"Recuperação de dados de um banco de dados" na página 757

"Inserção de dados" na página 767

"Alteração ou exclusão de dados" na página 773

Uso de parâmetros em instruções

Adobe AIR 1.0 e posterior

O uso de parâmetros de instrução SQL permite criar uma instrução SQL reutilizável. Quando você usa parâmetros de instrução, os valores na instrução podem mudar (como os que estão sendo adicionados a uma instrução `INSERT`), mas seu texto básico permanece inalterado. Consequentemente, usar parâmetros oferece o benefício do desempenho, além de facilitar a codificação de um aplicativo.

Noções básicas sobre parâmetros de instrução

Adobe AIR 1.0 e posterior

É frequente um aplicativo usar uma única instrução SQL várias vezes, com uma leve variação. Por exemplo, considere um aplicativo de controle de inventário no qual um usuário pode adicionar itens de inventário ao banco de dados. O código do aplicativo que adiciona um item de inventário ao banco de dados executa uma instrução SQL `INSERT` que, na verdade, adiciona os dados ao banco de dados. No entanto, cada vez que a instrução é executada, há uma leve variação. Especificamente, os valores reais inseridos na tabela são diferentes porque são específicos do item de inventário que está sendo adicionado.

Em casos nos quais você tem uma instrução SQL que é usada várias vezes com diferentes valores na instrução, a melhor abordagem é utilizar uma instrução SQL que inclua parâmetros em vez de valores literais no texto SQL. Um parâmetro consiste em um alocador de espaço no texto da instrução que é substituído por um valor real sempre que a instrução é executada. Para usar parâmetros em uma instrução SQL, crie a instância de SQLStatement como de costume. No caso da instrução SQL propriamente dita atribuída à propriedade `text`, use alocadores de espaço de parâmetro em vez de valores literais. Em seguida, defina o valor para cada parâmetro definindo o valor de um elemento na propriedade `parameters` da ocorrência de SQLStatement. A propriedade `parameters` é uma matriz associativa, por isso você define um valor em particular usando a seguinte sintaxe:

```
statement.parameters[parameter_identifier] = value;
```

parameter_identifier é uma string (se você está usando um parâmetro nomeado) ou um índice de inteiro (se está usando um parâmetro sem nome).

Uso de parâmetros nomeados

Adobe AIR 1.0 e posterior

Um parâmetro pode ser um parâmetro nomeado. Um parâmetro nomeado tem um nome específico que o banco de dados utiliza para comparar o valor do parâmetro à localização do alocador de espaço no texto da instrução. Um nome de parâmetro consiste no caractere ":" ou "@" seguido de um nome, como nestes exemplos:

```
:itemName
@firstName
```

A seguinte listagem de código demonstra o uso de parâmetros nomeados:

```
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();
```

Uso de parâmetros sem nome

Adobe AIR 1.0 e posterior

Como alternativa ao uso de parâmetros nomeados, também é possível utilizar parâmetros sem nome. Para usar um parâmetro sem nome, indique um parâmetro em uma instrução SQL usando um caractere "?" exigido. Cada parâmetro recebe um índice numérico conforme a ordem dos parâmetros da instrução, começando com o índice 0 para o primeiro parâmetro. Este exemplo demonstra uma versão do exemplo anterior usando parâmetros sem nome:

```
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();
```

Benefícios do uso de parâmetros

Adobe AIR 1.0 e posterior

O uso de parâmetros em uma instrução SQL proporciona vários benefícios:

Melhor desempenho Uma ocorrência de SQLStatement que usa parâmetros é executada com mais eficiência se comparada a uma que cria o texto SQL dinamicamente sempre que é executada. A melhoria do desempenho ocorre porque a instrução é preparada uma única vez e pode ser executada várias vezes usando-se diferentes valores de parâmetro, sem a necessidade de recompilar a instrução SQL.

Digitação de dados explícita Os parâmetros são usados para permitir a substituição de valores digitados que são desconhecidos no momento em que a instrução SQL é construída. O uso de parâmetros é o único modo de garantir a classe de armazenamento para um valor passado ao banco de dados. Quando parâmetros não são usados, o tempo de execução tenta converter todos os valores de sua representação em texto para uma classe de armazenamento com base na afinidade de tipo da coluna associada.

Para obter mais informações sobre classes de armazenamento e afinidade de coluna, consulte "Suporte ao tipo de dados" na página 1171.

Mais segurança O uso de parâmetros ajuda a impedir uma técnica mal-intencionada conhecida como ataque de injeção SQL. Em um ataque de injeção SQL, um usuário insere o código SQL em um local acessível ao usuário (por exemplo, um campo de entrada de dados). Se o código do aplicativo construir uma instrução SQL concatenando diretamente a entrada do usuário no texto SQL, o código SQL inserido pelo usuário será executado em relação ao banco de dados. A lista a seguir mostra um exemplo de concatenação da entrada do usuário no texto SQL. **Não use esta técnica**:

```
// 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;
```

O uso de parâmetros de instrução em vez de concatenar valores inseridos pelo usuário no texto de uma instrução impede o ataque de injeção SQL. A injeção de SQL não ocorre porque os valores de parâmetro são tratados explicitamente como valores substituídos, em vez de se tornarem parte do texto da instrução literal. Esta é uma alternativa recomendável à listagem anterior:

```
// 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;
```

Recuperação de dados de um banco de dados

Adobe AIR 1.0 e posterior

A recuperação de dados de um banco de dados envolve duas etapas. Primeiro, você executa uma instrução SQL `SELECT` que descreve o conjunto de dados desejado do banco de dados. Em seguida, você acessa os dados recuperados e os exibe ou manipula conforme exigido pelo aplicativo.

Execução de uma instrução SELECT

Adobe AIR 1.0 e posterior

Para recuperar dados existentes de um banco de dados, use uma instância de SQLStatement. Atribua a instrução SQL `SELECT` adequada à propriedade `text` da ocorrência e, depois, chame o método `execute()` correspondente.

Para obter mais detalhes sobre a sintaxe da instrução `SELECT`, consulte “Suporte SQL em bancos de dados locais” na página 1148.

O exemplo abaixo mostra como executar uma instrução `SELECT` para recuperar dados de uma tabela chamada “products” usando o modo de execução assíncrona:

```
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>
```

O exemplo a seguir demonstra a execução de uma instrução `SELECT` para recuperar dados de uma tabela chamada “produtos”, usando o modo de execução síncrona:

```
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>
```

No modo de execução assíncrona, quando a execução da instrução é concluída, a ocorrência de SQLStatement despacha um evento `result` (`SQLEvent.RESULT`) que indica que a instrução foi executada com sucesso. Como alternativa, se um objeto Responder for passado como argumento no método de `execute()`, a função do manipulador resultante do objeto Responder será chamada. No modo de execução síncrona, a execução pausa até que a operação `execute()` seja concluída e, em seguida, prossegue com a próxima linha de código.

Acesso a dados de resultados da instrução SELECT

Adobe AIR 1.0 e posterior

Uma vez concluída a execução da instrução `SELECT`, a próxima etapa é acessar os dados que foram recuperados. Acesse os dados de resultados com a execução de uma instrução `SELECT`, chamando o método `getResult()` do objeto SQLStatement:

```
var result:SQLResult = selectStatement.getResult();
```

O método `getResult()` gera um objeto SQLResult. A propriedade `data` do objeto SQLResult é um Array que contém os resultados da instrução `SELECT`:

```
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];
}
```

Cada linha de dados no conjunto de resultados de `SELECT` se transforma em uma instância do Objeto contida no Array `data`. Esse objeto tem propriedades cujos nomes correspondem aos nomes de coluna do conjunto de resultados. As propriedades contêm os valores das colunas do conjunto de resultados. Por exemplo, suponha que uma instrução `SELECT` especifique um conjunto de resultados com três colunas chamadas “itemId”, “itemName” e “price”. Para cada linha do conjunto de resultados, é criada uma ocorrência de Object com propriedades chamadas `itemId`, `itemName` e `price`. Essas propriedades contêm os valores das respectivas colunas.

A listagem de código a seguir define uma ocorrência de SQLStatement cujo texto é uma instrução `SELECT`. A instrução recupera linhas que contêm os valores de coluna `firstName` e `lastName` de todas as linhas de uma tabela denominada `employees`. Este exemplo utiliza o modo de execução assíncrona. Quando a execução é concluída, o método `selectResult()` é chamado, e as linhas de dados resultantes são acessadas usando `SQLStatement.getResult()` e exibidas usando o método `trace()`. Observe que esta listagem presume que existe uma ocorrência de SQLConnection denominada `conn` que já foi instanciada e já está conectada ao banco de dados. Ela também presume que a tabela "employees" já foi criada e preenchida com dados.

```
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>
```

A listagem de código a seguir demonstra as mesmas técnicas que a anterior, mas usa o modo de execução síncrona. O exemplo define uma instância de SQLStatement cujo texto é uma instrução `SELECT`. A instrução recupera linhas que contêm os valores de coluna `firstName` e `lastName` de todas as linhas de uma tabela denominada `employees`. As linhas de dados resultantes são acessadas usando `SQLStatement.getResult()` e exibidas usando o método `trace()`. Observe que esta listagem presume que existe uma ocorrência de SQLConnection denominada `conn` que já foi instanciada e já está conectada ao banco de dados. Ela também presume que a tabela "employees" já foi criada e preenchida com dados.

```
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>
```

Definição do tipo de dados dos dados resultantes de SELECT

Adobe AIR 1.0 e posterior

Por padrão, cada linha retornada por uma instrução `SELECT` é criada como uma ocorrência de Object com propriedades nomeadas para os nomes de coluna do conjunto de resultados e com o valor de cada coluna como o valor da propriedade associada. Porém, antes de executar uma instrução SQL `SELECT`, você pode definir a propriedade `itemClass` da ocorrência de SQLStatement para uma classe. Quando é definida a propriedade `itemClass`, cada linha retornada pela instrução `SELECT` é criada como ocorrência da classe designada. O tempo de execução atribui valores de coluna resultantes a valores de propriedade comparando os nomes de coluna do conjunto de resultados de `SELECT` com os nomes das propriedades da classe `itemClass`.

Qualquer classe designada como um valor da propriedade `itemClass` deve ter um construtor que não exija parâmetros. Além disso, a classe deve ter uma única propriedade para cada coluna retornada pela instrução `SELECT`. Será considerado um erro se uma coluna da lista `SELECT` não tiver um nome de propriedade correspondente na classe `itemClass`.

Recuperação dos resultados de SELECT em partes

Adobe AIR 1.0 e posterior

Por padrão, a execução de uma instrução `SELECT` recupera todas as linhas do conjunto de resultados de uma só vez. Uma vez concluída a instrução, geralmente você processa os dados recuperados de alguma forma; por exemplo, criando objetos ou exibindo os dados na tela. Se a instrução retornar muitas linhas, processar todos os dados de uma só vez pode exigir bastante do computador, o que, por sua vez, faz com que a interface do usuário não se redesenhe.

É possível melhorar o desempenho percebido do aplicativo instruindo o tempo de execução a retornar um determinado número de linhas de resultado por vez. Isso faz com que os dados resultantes iniciais sejam retornados mais rapidamente. Isso também permite dividir as linhas de resultados em conjuntos, para que a interface de usuário seja atualizada depois do processamento de cada conjunto de linhas. Só é prático usar esta técnica no modo de execução assíncrona.

Para recuperar os resultados de `SELECT` em partes, especifique um valor para o primeiro parâmetro do método `SQLStatement.execute()` (o parâmetro `prefetch`). O parâmetro `prefetch` indica o número de linhas a serem recuperadas na primeira vez que a instrução é executada. Quando chamar o método `execute()` de uma instância de SQLStatement, especifique um valor de parâmetro `prefetch` e somente essas linhas serão recuperadas:

```
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
```

A instrução despacha o evento `result`, indicando que o primeiro conjunto de linhas de resultado está disponível. A propriedade `data` da instância de SQLResult resultante contém as linhas de dados, e sua propriedade `complete` indica se existem mais linhas de resultado a serem recuperadas. Para recuperar linhas de resultados adicionais, chame o método `next()` da ocorrência de SQLStatement. Assim como o método `execute()`, o primeiro parâmetro do método `next()` é usado para indicar quantas linhas deverão ser recuperadas na próxima vez que o evento result for despachado.

```
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);
        }
    }
}
```

SQLStatement despacha um evento `result` sempre que o método `next()` retorna um conjunto subsequente de linhas de resultado. Consequentemente, a mesma função de ouvinte pode ser usada para continuar processando os resultados (de chamadas de `next()`) até que todas as linhas sejam recuperadas.

Para obter mais informações, consulte as descrições do método `SQLStatement.execute()` (a descrição do parâmetro `prefetch`) e o método `SQLStatement.next()`.

Inserção de dados

Adobe AIR 1.0 e posterior

A inclusão de dados em um banco de dados envolve executar uma instrução SQL `INSERT`. Depois de concluída a execução da instrução, você poderá acessar a chave primária da linha recém-inserida se a chave tiver sido gerada pelo banco de dados.

Execução de uma instrução INSERT

Adobe AIR 1.0 e posterior

Para adicionar dados a uma tabela de um banco de dados, crie e execute uma instância de SQLStatement cujo texto é uma instrução SQL `INSERT`.

O exemplo a seguir usa uma ocorrência de SQLStatement para adicionar uma linha de dados à tabela employees já existente. Este exemplo demonstra como inserir dados usando o modo de execução assíncrona. Observe que esta listagem presume que existe uma instância de SQLConnection denominada `conn` que já foi instanciada e já está conectada a um banco de dados. Ela também presume que a tabela “employees” já foi criada.

```
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>
```

O exemplo a seguir adiciona uma linha de dados à tabela employees já existente usando o modo de execução síncrona. Observe que esta listagem presume que existe uma instância de SQLConnection denominada `conn` que já foi instanciada e já está conectada a um banco de dados. Ela também presume que a tabela “employees” já foi criada.

```
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>
```

Recuperação de uma chave primária gerada pelo banco de dados de uma linha inserida

Adobe AIR 1.0 e posterior

Com frequência, depois de inserir uma linha de dados a uma tabela, o código precisa ser informado sobre uma chave primária gerada pelo banco de dados ou o valor do identificador da linha recém-inserida. Por exemplo, depois de inserir uma linha em uma tabela, você deve adicionar linhas a uma tabela relacionada. Nesse caso, convém inserir o valor da chave primária como chave externa na tabela relacionada. A chave primária de uma linha recém-inserida pode ser recuperada usando o objeto SQLResult associado à execução da instrução. É o mesmo objeto usado para acessar dados de resultado depois que uma instrução `SELECT` é executada. Assim como ocorre com qualquer instrução SQL, quando a execução de uma instrução `INSERT` é concluída, o tempo de execução cria uma ocorrência de SQLResult. Para acessar a instância de SQLResult, chame o método `getResult()` do objeto SQLStatement se você estiver usando

um ouvinte de evento ou o modo de execução síncrona. Se preferir, caso esteja usando o modo de execução assíncrona e passe uma instância de Responder para a chamada de `execute()`, a instância de SQLResult será passada como argumento para a função do manipulador resultante. Seja como for, a ocorrência de SQLResult tem uma propriedade, chamada `lastInsertRowID`, que contém o identificador de linha da linha inserida mais recentemente se a instrução SQL executada é uma instrução `INSERT`.

Este exemplo mostra como acessar a chave primária de uma linha inserida no modo de execução assíncrona:

```
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
}
```

Este exemplo mostra como acessar a chave primária de uma linha inserida no modo de execução síncrona:

```
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
}
```

Observe que o identificador de linha pode ou não ser o valor da coluna designada como a coluna de chave primária na definição de tabela, de acordo com as seguintes regras:

- Se a tabela está definida com uma coluna de chave primária cuja afinidade (tipo de dados da coluna) é `INTEGER`, a propriedade `lastInsertRowID` contém o valor inserido nessa linha (ou o valor gerado pelo tempo de execução, no caso de uma coluna `AUTOINCREMENT`).
- Se a tabela estiver definida com várias colunas de chave primária (uma chave composta) ou com uma única coluna de chave primária cuja afinidade não é `INTEGER`, o banco de dados gera um valor inteiro identificador de coluna para a linha em segundo plano. Esse valor gerado é o valor da propriedade `lastInsertRowID`.
- O valor é sempre o identificador de linha da linha inserida mais recentemente. Se uma instrução `INSERT` faz com que seja acionado um disparador que, por sua vez, insere uma linha, a propriedade `lastInsertRowID` contém o identificador de linha da última linha inserida pelo disparador em vez da linha criada pela instrução `INSERT`.

Em virtude dessas regras, se você quiser ter uma coluna de chave primária definida explicitamente cujo valor fica disponível após um comando `INSERT` através da propriedade `SQLResult.lastInsertRowID`, a coluna deverá ser definida como uma coluna `INTEGER PRIMARY KEY`. Mesmo que a tabela não inclua uma coluna `INTEGER PRIMARY KEY` explícita, será igualmente aceitável usar o identificador de linha gerado pelo banco de dados como uma chave primária para a sua tabela com o intuito de definir relacionamentos com tabelas relacionadas. O valor da coluna do identificador de linha fica disponível em qualquer instrução SQL através do uso de um dos nomes de coluna especiais `ROWID`, `_ROWID_` ou `OID`. É possível criar uma coluna de chave externa em uma tabela relacionada e usar o valor do identificador de linha como o valor da coluna de chave externa, assim como você faria com uma coluna `INTEGER PRIMARY KEY` declarada explicitamente. Nesse sentido, se estiver usando uma chave primária arbitrária em vez de uma chave natural, e desde que não se importe que o tempo de execução gere o valor da chave primária para você, faz pouca diferença se você usa uma coluna `INTEGER PRIMARY KEY` ou o identificador de linha gerado pelo sistema como a chave primária de uma tabela para definir um relacionamento de chave externa entre duas tabelas.

Para obter mais detalhes sobre chaves primárias e identificadores e linha gerados, consulte "Suporte SQL em bancos de dados locais" na página 1148.

Alteração ou exclusão de dados

Adobe AIR 1.0 e posterior

O processo para executar outras operações de manipulação de dados é idêntico ao seguido para executar uma instrução `SQL SELECT` ou `INSERT`, conforme descrito em "Trabalhar com instruções SQL" na página 753. Basta substituir uma instrução SQL diferente na propriedade `text` da instância de SQLStatement:

- Para alterar dados existentes em uma tabela, use uma instrução `UPDATE`.
- Para excluir uma ou mais linhas de dados de uma tabela, use uma instrução `DELETE`.

Para obter descrições dessas instruções, consulte "Suporte SQL em bancos de dados locais" na página 1148.

Trabalhar com vários bancos de dados

Adobe AIR 1.0 e posterior

Use o método `SQLConnection.attach()` para abrir uma conexão com um banco de dados adicional em uma ocorrência de SQLConnection que já tem um banco de dados aberto. Dê um nome ao banco de dados anexado usando o parâmetro de nome na chamada do método `attach()`. Ao escrever instruções para manipular esse banco de dados, você poderá usar esse nome em um prefixo (no formato `database-name.table-name`) para qualificar qualquer nome de tabela em suas instruções SQL, indicando ao tempo de execução que a tabela pode ser encontrada no banco de dados nomeado.

É possível executar uma única instrução SQL que inclua tabelas de vários bancos de dados conectados à mesma ocorrência de SQLConnection. Se uma transação for criada na ocorrência de SQLConnection, ela será aplicada a todas as instruções SQL executadas usando a ocorrência de SQLConnection. Isso é válido independentemente do banco de dados anexado no qual a instrução é executada.

Se preferir, você também pode criar várias ocorrências de SQLConnection em um aplicativo, cada uma delas conectada a um ou a vários bancos de dados. Contudo, se você usar várias conexões com o mesmo banco de dados, lembre-se de que uma transação de banco de dados não é compartilhada entre ocorrências de SQLConnection. Consequentemente, se você se conectar ao mesmo arquivo de banco de dados usando várias ocorrências de SQLConnection, não poderá contar que as alterações de dados em ambas as conexões sejam aplicadas da maneira esperada. Por exemplo, se duas instruções `UPDATE` ou `DELETE` forem executadas no mesmo banco de dados através de diferentes ocorrências de SQLConnection e acontecer um erro de aplicativo depois de executada uma operação, os dados do banco de dados poderão ficar em um estado intermediário irreversível que talvez afete a integridade do banco de dados (e, consequentemente, o aplicativo).

Tratamento de erros de banco de dados

Adobe AIR 1.0 e posterior

Em geral, o tratamento de erros de banco de dados é parecido com o de outros erros de tempo de execução. Você deve criar um código preparado para eventuais erros e para responder a eles em vez de deixar que o tempo de execução faça isso. De um modo geral, os erros de banco de dados possíveis dividem-se em três categorias: erros de conexão, de sintaxe SQL e de restrição.

Erro de conexão

Adobe AIR 1.0 e posterior

A maioria dos erros de banco de dados são de conexão, e eles podem ocorrer durante qualquer operação. Embora haja estratégias para prevenir erros de conexão, raramente existe uma forma simples de se recuperar tranquilamente de um erro desse tipo se o banco de dados é um componente crítico do seu aplicativo.

A maior parte dos erros de conexão tem a ver com a maneira como o tempo de execução interage com o sistema operacional, o sistema de arquivos e o arquivo de banco de dados. Por exemplo, ocorre um erro de conexão quando o usuário não tem permissão para criar um arquivo de banco de dados em um determinado local do sistema de arquivos. As seguintes estratégias ajudam a prevenir erros de conexão:

Utilize arquivos de banco de dados específicos do usuário Em vez de usar um único arquivo de banco de dados para todos os usuários que utilizam um mesmo computador, dê a cada usuário seu próprio arquivo de banco de dados. O arquivo deve estar localizado em um diretório associado com a conta do usuário. Por exemplo, ele pode estar no diretório de armazenamento do aplicativo, na pasta de documentos do usuário, na área de trabalho dele, e assim por diante.

Leve em consideração diferentes tipos de usuário Teste seu aplicativo com diferentes tipos de contas de usuário em diferentes sistemas operacionais. Não suponha que o usuário tem permissão de administrador no computador. Também não presuma que quem instalou o aplicativo é o usuário que o está executando.

Leve em consideração localizações de arquivo diferentes Se você permitir que um usuário especifique onde salvar um arquivo de banco de dados ou selecione um arquivo a ser aberto, considere as possíveis localizações de arquivo que os usuários podem usar. Além disso, considere a possibilidade de definir limites quanto aos locais em que os usuários podem armazenar (ou abrir) arquivos de banco de dados. Por exemplo, você só deve permitir que eles abram arquivos existentes no local de armazenamento de suas contas de usuário.

Se acontecer um erro de conexão, é mais provável que ele ocorra na primeira tentativa de criar ou abrir o banco de dados. Isso significa que o usuário não consegue fazer nenhuma operação relacionada a banco de dados no aplicativo. Para certos tipos de erros, como erros de permissão ou somente leitura, uma técnica de recuperação possível consiste em copiar o arquivo de banco de dados para outro local. O aplicativo pode copiar o arquivo de banco de dados para outro local, em que o usuário tenha permissão para criar e gravar em arquivos, e pode usar esse local.

Erros de sintaxe

Adobe AIR 1.0 e posterior

Um erro de sintaxe ocorre quando uma instrução SQL está formada incorretamente e o aplicativo tenta executá-la. Como as instruções SQL de banco de dados local são criadas como strings, não é possível fazer a verificação da sintaxe SQL durante a compilação. Todas as instruções SQL devem ser executadas para verificar sua sintaxe. Adote as seguintes estratégias para evitar erros de sintaxe SQL:

Teste todas as instruções SQL integralmente Se possível, enquanto desenvolve o aplicativo, teste as instruções SQL separadamente antes de codificá-las como texto de instrução no código do aplicativo. Além disso, use uma abordagem de teste do código, como teste de unidade, para criar um conjunto de testes que aplique todas as opções e variações possíveis no código.

Use parâmetros de instrução e evite concatenar (gerar dinamicamente) o SQL Usar parâmetros, e evitar instruções SQL criadas dinamicamente, significa que o mesmo texto de instrução SQL é usado sempre que uma instrução é executada. Consequentemente, é bem mais fácil testar suas instruções e limitar a variação possível. Se você tiver de gerar uma instrução SQL dinamicamente, use o mínimo possível de partes dinâmicas na instrução. Além disso, seja cauteloso ao validar qualquer entrada de usuário para se certificar de que ela não causará erros de sintaxe.

Para se recuperar de um erro de sintaxe, um aplicativo precisa de uma lógica complexa que permita examinar uma instrução SQL e corrigir sua sintaxe. Se forem seguidas as diretrizes acima para evitar erros de sintaxe, o código conseguirá identificar qualquer possível origem de tempo de execução de erros de sintaxe SQL (como entradas de usuário utilizadas em uma instrução). Para se recuperar de um erro de sintaxe, oriente o usuário. Indique o que deve ser corrigido para que a instrução seja executada corretamente.

Erros de restrição

Adobe AIR 1.0 e posterior

Erros de restrição ocorrem quando uma instrução `INSERT` ou `UPDATE` tenta adicionar dados a uma coluna. O erro acontece se os novos dados violam uma das restrições definidas para a tabela ou coluna. O conjunto de restrições possíveis inclui:

Restrição exclusiva Indica que, entre todas as linhas de uma tabela, não pode existir valores duplicados em uma coluna. Como alternativa, quando várias colunas são combinadas em uma restrição exclusiva, a combinação de valores dessas colunas não deve ser duplicada. Em outras palavras, em termos da(s) coluna(s) exclusiva(s) especificada(s), cada linha deve ser distinta.

Restrição de chave primária Em termos dos dados permitidos e não permitidos por uma restrição, a restrição de chave primária é idêntica a uma restrição exclusiva.

Restrição não nula Especifica que uma única coluna não pode armazenar um valor `NULL` e, consequentemente, que em cada linha essa coluna deve ter um valor.

Restrição de verificação Permite especificar uma restrição arbitrária em uma ou mais tabelas. Uma restrição de verificação comum é uma regra que define que o valor de uma coluna deve estar dentro de certos limites (por exemplo, que o valor de uma coluna numérica deve ser maior que 0). Outro tipo de restrição de verificação comum especifica relacionamentos entre valores de coluna (por exemplo, que o valor de uma coluna deve ser diferente do de outra coluna na mesma linha).

Restrição de tipo de dados (afinidade de coluna) O tempo de execução impõe o tipo de dados de valores de colunas, e ocorre um erro se é feita uma tentativa de armazenar um valor de tipo incorreto em uma coluna. No entanto, em muitas condições os valores são convertidos para corresponder ao tipo de dados declarado da coluna. Consulte "Trabalhar com tipos de dados de banco de dados" na página 777 para obter mais informações.

O tempo de execução não impõe restrições quanto a valores de chave externa. Em outras palavras, os valores de chave externa não precisam corresponder ao valor de uma chave primária existente.

Além dos tipos de restrição predefinidos, o mecanismo SQL de tempo de execução permite o uso de disparadores. O disparador é como um manipulador de eventos. Ele consiste em um conjunto de instruções predefinidas que são executadas quando acontece uma determinada ação. Por exemplo, é possível definir um disparador que seja executado quando dados forem inseridos ou excluídos de uma tabela específica. Um uso possível de um disparador é examinar alterações feitas em dados e fazer com que ocorra um erro se as condições especificadas não forem atendidas. Consequentemente, um disparador pode atender ao mesmo objetivo de uma restrição, e as estratégias para evitar e se recuperar de erros de restrição também se aplicam a erros gerados por disparadores. Porém, a id de erro dos erros gerados por disparadores é diferente da id de erro dos erros de restrição.

O conjunto de restrições que se aplicam a uma tabela em particular é determinado enquanto você está criando um aplicativo. Elaborar restrições com consciência facilita que você crie o aplicativo para evitar e se recuperar de erros de constrição. No entanto, os erros de restrição são difíceis de prever e impedir sistematicamente. A previsão é difícil porque os erros de restrição só aparecem depois que são adicionados os dados do aplicativo. Os erros de restrição acontecem com dados que são adicionados a um banco de dados após sua criação. Com frequência, esses erros são resultado do relacionamento entre os novos dados e os dados já existentes no banco de dados. As seguintes estratégias podem ajudá-lo a evitar muitos erros de restrição:

Planeje cautelosamente a estrutura e as restrições do banco de dados A finalidade das restrições é impor regras de aplicativo e ajudar a proteger a integridade dos dados do banco de dados. Quando estiver planejando seu aplicativo, pense em como estruturar o banco de dados para dar suporte ao aplicativo. Como parte desse processo, identifique regras para seus dados; por exemplo, se certos valores não necessários, se um valor tem um padrão, se são permitidos valores duplicados e assim por diante. Essas regras orientam a definição de restrições do banco de dados.

Especifique nomes de coluna explicitamente É possível criar uma instrução `INSERT` sem especificar explicitamente as colunas em que os valores deverão ser inseridos, mas fazê-lo é um risco desnecessário. Quando você nomeia explicitamente as colunas em que os valores deverão ser inseridos, pode permitir valores gerados de forma automática, colunas com valores padrão e colunas que permitam valores `NULL`. Além disso, quando você faz isso pode assegurar que todas as colunas `NOT NULL` terão um valor explícito inserido.

Use valores padrão Sempre que você especificar uma restrição `NOT NULL` para uma coluna, se possível, especifique um valor padrão na definição de coluna. O código do aplicativo também pode fornecer valores padrão. Por exemplo, o código pode verificar se uma variável String é `null` e atribuir um valor a ela antes de usá-la para definir um valor de parâmetro de instrução.

Valide os dados inseridos pelos usuários Verifique com antecedência os dados inseridos pelos usuários para ter certeza de que obedecem aos limites especificados pelas restrições, principalmente no caso de restrições `NOT NULL` e `CHECK`. Naturalmente, uma restrição `UNIQUE` é mais difícil de verificar porque, para fazer a verificação, é necessário executar uma consulta `SELECT` para determinar se os dados são exclusivos.

Use disparadores Você pode criar um disparador que valide (e possivelmente substitua) os dados inseridos ou que tome outras medidas para corrigir dados inválidos. Esse procedimento de validação e correção pode evitar que aconteça um erro de restrição.

Em vários aspectos, é mais difícil evitar os erros de restrição do que outros tipos de erro. Felizmente, existem diversas estratégias para se recuperar de erros de restrição sem tornar o aplicativo instável ou inutilizável:

Use algoritmos de conflito Quando você define uma restrição em uma coluna, e quando cria uma instrução `INSERT` ou `UPDATE`, tem a opção de especificar um algoritmo de conflito. Um algoritmo de conflito define a ação que o banco de dados executa quando ocorre uma violação de restrição. Há várias ações possíveis que o mecanismo de banco de dados pode executar. O mecanismo de banco de dados pode encerrar uma única instrução ou uma transação inteira. Ele pode ignorar o erro. Ele pode até mesmo remover dados antigos e substituí-los pelos dados que o código está tentando armazenar.

Para obter mais informações consulte a seção "ON CONFLICT (algoritmos de conflito)" em "Suporte SQL em bancos de dados locais" na página 1148.

Disponibilize feedback de correção É possível identificar antecipadamente o conjunto de restrições que pode afetar um determinado comando SQL. Como consequência, você consegue prever os erros de restrição que podem ser causados por uma instrução. Dispondo dessa informação, você pode desenvolver a lógica do aplicativo para responder a um erro de restrição. Por exemplo, suponha que um aplicativo inclui um formulário de entrada de dados que permite inserir novos produtos. Se a coluna de nome de produto no banco de dados estiver definida com uma restrição `UNIQUE`, a ação de inserir uma linha de novo produto no banco de dados poderá causar um erro de restrição. Consequentemente, o aplicativo deve prever um erro de restrição. Quando o erro acontece, o aplicativo alerta o usuário indicando que o nome de produto especificado já está sendo utilizado e pede a ele para escolher outro nome. Outra resposta possível é permitir que o usuário visualize informações sobre o outro produto que tem o mesmo nome.

Trabalhar com tipos de dados de banco de dados

Adobe AIR 1.0 e posterior

Quando uma tabela é criada em um banco de dados, a instrução SQL usada para criar a tabela define a afinidade, ou o tipo de dados, de cada coluna da tabela. Embora seja possível omitir declarações de afinidade, é uma boa ideia declarar explicitamente a afinidade de coluna nas instruções SQL `CREATE TABLE`.

Como regra geral, qualquer objeto que você armazena em um banco de dados usando uma instrução `INSERT` é retornado como ocorrência do mesmo tipo de dados quando é executada uma instrução `SELECT`. No entanto, o tipo de dados do valor recuperado pode ser diferente, dependendo da afinidade da coluna do banco de dados na qual o valor está armazenado. Quando um valor é armazenado em uma coluna, se o respectivo tipo de dados não corresponde à afinidade de coluna, o banco de dados tenta converter o valor para que corresponda à afinidade de coluna. Por exemplo, se uma coluna de banco de dados está declarada com afinidade `NUMERIC`, o banco de dados tenta converter os dados inseridos em uma classe de armazenamento numérica (`INTEGER` ou `REAL`) antes de armazenar os dados. O banco de dados gera um erro se não é possível converter os dados. De acordo com esta regra, se a String "12345" é inserida em uma coluna `NUMERIC`, o banco de dados automaticamente a converte no valor inteiro 12345 antes de armazená-la no banco de dados. Quando recuperado com uma instrução `SELECT`, o valor é retornado como uma ocorrência de um tipo de dados numérico (por exemplo, Number) e não como uma ocorrência de String.

A melhor maneira de evitar a conversão indesejada de tipos de dados é seguir duas regras. Primeiro, defina cada coluna com a afinidade que corresponda ao tipo de dados que pretende armazenar. Em seguida, insira apenas valores cujo tipo de dados corresponda à afinidade definida. Se você seguir essas regras, terá dois benefícios. Quando você inserir os dados, eles não serão convertidos inesperadamente (com a possibilidade de perderem o significado pretendido). Além disso, quando você recupera os dados, eles são retornados com o tipo de dados original.

Para obter mais informações sobre os tipos de afinidade de coluna disponíveis e utilização de tipos de dados em instruções SQL, consulte o "Suporte ao tipo de dados" na página 1171.

Uso de operações de banco de dados síncronas e assíncronas

Adobe AIR 1.0 e posterior

As seções anteriores descreveram operações comuns de banco de dados, como recuperar, inserir, atualizar e excluir dados, além da criação de um arquivo de banco de dados e de tabelas e outros objetos de um banco de dados. Os exemplos demonstraram como executar essas operações de maneira assíncrona e síncrona.

Como lembrete, no modo de execução assíncrona, você instrui o mecanismo de banco de dados a executar uma operação. O mecanismo de banco de dados então trabalha em segundo plano enquanto o aplicativo continua a executar. Quando a operação é concluída, o mecanismo de banco de dados despacha um evento para alertar você sobre o fato. O principal benefício da execução assíncrona é que o tempo de execução realiza as operações de banco de dados em segundo plano, enquanto o código do aplicativo principal continua a executar. Isso é importante principalmente quando a operação é bastante demorada.

Por outro lado, no modo de execução síncrona, as operações não são executadas em segundo plano. Você instrui o mecanismo de banco de dados a executar uma operação. O código pausa nesse ponto enquanto o mecanismo de banco de dados faz o seu trabalho. Quando a operação é concluída, a execução prossegue com a próxima linha do código que você criou.

Uma única conexão de banco de dados não pode executar algumas operações ou instruções em modo síncrono e outras em modo assíncrono. Especifique se uma SQLConnection opera de forma síncrona ou assíncrona quando abre a conexão com o banco de dados. Se você chamar `SQLConnection.open()`, a conexão funcionará no modo de execução síncrona; se você chamar `SQLConnection.openAsync()`, ela funcionará no modo de execução assíncrona. Uma vez que uma ocorrência de SQLConnection é conectada a um banco de dados usando `open()` ou `openAsync()`, ela é fixada à execução síncrona ou assíncrona.

Uso de operações de banco de dados síncronas

Adobe AIR 1.0 e posterior

Há pouca diferente no código que você usa para executar e responder a operações na execução síncrona em comparação com o código para o modo de execução assíncrona. As principais diferenças entre as duas abordagens estão em duas áreas. A primeira é executar uma operação que depende de outra operação (como linhas de resultado `SELECT` ou a chave primária da linha adicionada por uma instrução `INSERT`). A segunda área de diferença é o tratamento de erros.

Criação de código para operações síncronas

Adobe AIR 1.0 e posterior

A principal diferença entre a execução síncrona e assíncrona é que, no modo síncrono, você cria o código como uma única série de etapas. Por outro lado, no código assíncrono, você registra ouvintes de evento e com frequência divide as operações entre métodos de ouvinte. Quando um banco de dados é conectado no modo de execução síncrona, você pode executar uma série de operações de banco de dados em sequência em um único bloco de código. O exemplo abaixo demonstra esta técnica:

```
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();
```

Como você pode ver, é possível chamar os mesmos métodos para executar operações de banco de dados independentemente de usar a execução síncrona ou assíncrona. As principais diferenças entre as duas abordagens são executar uma operação que depende de outra operação e como tratar de erros.

Execução de uma operação que depende de outra

Adobe AIR 1.0 e posterior

Quando você usa o modo de execução síncrona, não precisa criar um código que monitore um evento para determinar quando uma operação é concluída. Em vez disso, você pode presumir que, se uma operação em uma linha de código foi concluída com sucesso, a execução continuará na próxima linha de código. Consequentemente, para executar uma operação que depende do sucesso de outra, basta criar o código dependente logo depois da operação da qual ela depende. Por exemplo, para codificar um aplicativo para iniciar uma transação, executar uma instrução `INSERT`, recuperar a chave primária da linha inserida, inserir essa chave primária em outra linha de uma outra tabela e, por fim, confirmar a transação, o código pode ser todo criado como uma série de instruções. O seguinte exemplo demonstra essas operações:

```
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();
```

Tratamento de erros na execução síncrona

Adobe AIR 1.0 e posterior

No modo de execução síncrona, você não monitora um evento de erro para determinar se uma operação falhou. Em vez disso, coloque qualquer código que possa disparar erros em um conjunto de blocos de código `try..catch..finally`. Coloque o código gerador de erro no bloco `try`. Grave as ações a serem executadas em resposta a cada tipo de erro em blocos `catch` separados. Coloque qualquer código que você queira sempre executar independentemente de sucesso ou falha (por exemplo, encerrar uma conexão de banco de dados não mais necessária) em um bloco `finally`. O exemplo a seguir demonstra o uso de blocos `try..catch..finally` para tratamento de erros. Ele aproveita o exemplo anterior adicionando código de tratamento de erro:

```
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();
}
```

Noções básicas sobre o modelo de execução assíncrona

Adobe AIR 1.0 e posterior

Uma preocupação comum sobre o uso do modo de execução assíncrona é a suposição de que não é possível começar a executar uma instância de SQLStatement se outra instância de SQLStatement está em execução na mesma conexão de banco de dados. Na verdade, essa suposição não está correta. Durante a execução de uma ocorrência de SQLStatement, não é possível alterar a propriedade `text` da instrução. Contudo, se você usar uma ocorrência de SQLStatement à parte para cada instrução SQL que deseja executar, poderá chamar o método `execute()` de um SQLStatement enquanto outra ocorrência de SQLStatement ainda estiver sendo executada, sem causar erro.

Internamente, quando você executa operações de banco de dados utilizando o modo de execução assíncrona, cada conexão de banco de dados (cada instância de SQLConnection) tem sua própria fila ou lista de operações que está instruída a executar. O tempo de execução realiza cada operação em sequência, na ordem em que são adicionadas à fila. Quando você cria uma ocorrência de SQLStatement e chama o método `execute()` relacionado, a operação de execução dessa instrução é adicionada à fila da conexão. Se nenhuma operação estiver sendo executada nessa ocorrência de SQLConnection, a instrução começará a ser executada em segundo plano. Suponha que, dentro do mesmo bloco de código, você crie outra ocorrência de SQLStatement e também chame o método `execute()` daquele método. A operação que executa essa segunda instrução é adicionada à fila depois da primeira instrução. Assim que é concluída a execução da primeira instrução, o tempo de execução passa para a próxima operação da fila. O processamento das operações subsequentes na fila acontece em segundo plano, mesmo quando o evento `result` da primeira operação está sendo despachado no código do aplicativo principal. O código abaixo demonstra esta técnica:

```
// 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.
```

Ocorre um importante efeito colateral pelo fato de o banco de dados executar as próximas instruções em fila de maneira automática. Se uma instrução depende do resultado de outra operação, não é possível adicioná-la à fila (em outras palavras, você não pode chamar o método `execute()` relacionado) até a primeira operação ser concluída. Isso acontece porque, depois que você chamou o método `execute()` da segunda instrução, não é possível alterar as propriedades `text` ou `parameters` dela. Nesse caso, você deve aguardar o evento indicando que a primeira operação foi concluída antes de iniciar a próxima operação. Por exemplo, se você quiser executar uma instrução no contexto de uma transação, a execução da instrução dependerá da operação de abrir a transação. Após chamar o método `SQLConnection.begin()` para abrir a transação, você terá de aguardar que a ocorrência de SQLConnection despache seu evento `begin`. Só então você poderá chamar o método `execute()` da ocorrência de SQLStatement. Neste exemplo, o modo mais simples de organizar o aplicativo para assegurar que as operações sejam executadas corretamente é criar um método que seja registrado como ouvinte do evento `begin`. O código para chamar o método `SQLStatement.execute()` é colocado dentro do método desse ouvinte.

Uso da criptografia com bancos de dados SQL

Adobe AIR 1.5 e posterior

Todos os aplicativos do Adobe AIR compartilham o mesmo mecanismo do banco de dados local. Consequentemente, qualquer aplicativo do AIR pode se conectar, ler e gravar em qualquer arquivo do banco de dados não criptografado. Começando com o Adobe AIR 1.5, o AIR inclui a capacidade de criar e se conectar a arquivos do banco de dados criptografado. Quando você usa um banco de dados criptografado, para se conectar a ele um aplicativo deve fornecer a chave de criptografia correta. Se a chave de criptografia incorreta (ou nenhuma chave) for fornecida, o aplicativo não poderá se conectar ao banco de dados. Consequentemente, o aplicativo não poderá ler, gravar ou alterar dados do banco de dados.

Para usar um banco de dados criptografado, você deve criar o banco de dados como banco de dados criptografado. Com um banco de dados criptografado existente, é possível abrir uma conexão com o banco de dados. Você também pode alterar a chave de criptografia de um banco de dados criptografado. Em vez de criar e estabelecer conexão com bancos de dados criptografados, as técnicas para trabalhar com um banco de dados criptografado são as mesmas que para um banco de dados não criptografado. Especificamente, a execução de instruções SQL ocorre da mesma forma em bancos de dados criptografados e não criptografados.

Uso de um banco de dados criptografado

Adobe AIR 1.5 e posterior

A criptografia é útil sempre que você desejar restringir o acesso às informações armazenadas em um banco de dados. A funcionalidade de criptografia do banco de dados do Adobe AIR pode ser usada para várias finalidades. Estes são exemplos de casos em que você pode usar um banco de dados criptografado:

- Um cache somente leitura de dados de aplicativos privados baixados de um servidor
- Um armazenamento de aplicativo local para dados privados sincronizado com um servidor (os dados são enviados e carregados do servidor)
- Arquivos criptografados usados como formato de arquivo para documentos criados e editados pelo aplicativo. Os arquivos podem ser privados para um usuário ou projetados para compartilhamento entre todos os usuários do aplicativo.
- Qualquer outro uso de um armazenamento local de dados, como os descrito em "Usos de bancos de dados SQL locais" na página 737, onde os dados devem ser mantidos privados de pessoas com acesso ao computador ou aos arquivos do banco de dados.

Compreender o motivo pelo qual você deseja usar um banco de dados criptografado ajuda a decidir como arquitetar seu aplicativo. Especificamente, ele pode afetar a forma que seu aplicativo cria, obtém e armazena a chave de criptografia para o banco de dados. Para obter mais informações sobre essas considerações, consulte "Considerações para usar criptografia com um banco de dados" na página 787.

Além de um banco de dados criptografado, um mecanismo alternativo para manter dados confidenciais privados é o armazenamento local criptografado. Com o armazenamento local criptografado, você armazena um único valor ByteArray usando uma chave String. Apenas o aplicativo do AIR que armazena o valor pode acessá-lo, e somente no computador em que ele está armazenado. Com o armazenamento local criptografado, não é necessário criar sua própria chave de criptografia. Por esses motivos, o armazenamento local criptografado é mais adequado para armazenar facilmente um único valor ou um conjunto de valores que possa ser facilmente codificado em um ByteArray. Um banco de dados criptografado é mais adequado para conjuntos de dados maiores, onde o armazenamento de dados e consultas estruturadas são desejáveis. Para obter mais informações sobre o uso de armazenamento local criptografado, consulte "Armazenamento local criptografado" na página 733.

Criação de um banco de dados criptografado

Adobe AIR 1.5 e posterior

Para usar um banco de dados criptografado, o arquivo do banco de dados deve ser criptografado quando ele for criado. Quando um banco de dados é criado como não criptografado, ele pode não ser criptografado posteriormente. Da mesma forma, um banco de dados criptografado não pode passar a não criptografado posteriormente. Se necessário, você também pode alterar a chave de criptografia de um banco de dados criptografado. Para obter detalhes, consulte "Alteração da chave de criptografia de um banco de dados" na página 786. Se você tiver um banco de dados existente que não seja criptografado e desejar usar a criptografia de banco de dados, poderá criar um novo banco de dados criptografado e copiar a estrutura de tabela e os dados existentes no novo banco de dados.

Criar um banco de dados criptografado é quase idêntico a criar um banco de dados não criptografado, como descrito em "Criação de um banco de dados" na página 742. Primeiro você cria uma instância do SQLConnection que representa a conexão ao banco de dados. Crie o banco de dados chamando o método `open()` ou o método `openAsync()` do objeto do SQLConnection, especificando para a localização do banco de dados um arquivo que ainda não exista. A única diferença na criação de um banco de dados criptografado é que você fornece um valor para o parâmetro `encryptionKey` (o quinto parâmetro do método `open()` e o sexto parâmetro do método `openAsync()`).

Um valor válido do parâmetro `encryptionKey` é um objeto ByteArray que contém exatamente 16 bytes.

Os exemplos a seguir demonstram a criação de um banco de dados criptografado. Para simplicidade, nestes exemplos, a chave de criptografia é codificada no código do aplicativo. No entanto, essa técnica é altamente desencorajada, pois não é segura.

```
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);
```

Para obter um exemplo que demonstre uma forma recomendada de gerar uma chave de criptografia, consulte "Exemplo: Geração e uso de uma chave de criptografia" na página 788.

Conexão com um banco de dados criptografado

Adobe AIR 1.5 e posterior

Da mesma forma que criar um banco de dados criptografado, o procedimento para abrir uma conexão com um banco de dados criptografado é como conectar-se a um banco de dados não criptografado. Esse procedimento é descrito com mais detalhes em "Conexão com um banco de dados" na página 750. Use o método `open()` para abrir uma conexão no modo de execução síncrona ou o método `openAsync()` para abri-la no modo de execução assíncrona. A única diferença é que para abrir um banco de dados criptografado, especifica o valor correto para o parâmetro `encryptionKey` (o quinto parâmetro do método `open()` e o sexto parâmetro do método `openAsync()`).

Se a chave de criptografia fornecida não for a correta, ocorre um erro. Para o método `open()`, é lançada uma exceção SQLError. Para o método `openAsync()`, o objeto SQLConnection gera um SQLErrorEvent, cuja propriedade `error` contém um objeto SQLError. Em qualquer um dos casos, o objeto SQLError gerado pela exceção tem o valor 3138 na propriedade `errorID`. Este ID de erro corresponde à mensagem de erro "O arquivo aberto não corresponde a um arquivo de banco de dados."

O exemplo a seguir demonstra a abertura de um banco de dados criptografado em modo de execução assíncrona. Para simplicidade, neste exemplo a chave de criptografia é codificada no código do aplicativo. No entanto, essa técnica é altamente desencorajada, pois não é segura.

```
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);
    }
}
```

O exemplo a seguir demonstra a abertura de um banco de dados criptografado em modo de execução síncrona. Para simplicidade, neste exemplo a chave de criptografia é codificada no código do aplicativo. No entanto, essa técnica é altamente desencorajada, pois não é segura.

```
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);
    }
}
```

Para obter um exemplo que demonstre uma forma recomendada de gerar uma chave de criptografia, consulte "Exemplo: Geração e uso de uma chave de criptografia" na página 788.

Alteração da chave de criptografia de um banco de dados

Adobe AIR 1.5 e posterior

Quando um banco de dados é criptografado, você pode alterar a chave de criptografia do banco de dados posteriormente. Para alterar a chave criptográfica de um banco de dados, primeiro abra uma conexão com o banco de dados criando uma instância de SQLConnection e chamando seu método `open()` ou `openAsync()`. Quando o banco de dados for conectado, chame o método `reencrypt()`, enviando a nova chave de criptografia como um argumento.

Como a maioria das operações de banco de dados, o comportamento do método `reencrypt()` depende se a conexão do banco de dados usa modo de execução síncrona ou assíncrona. Se você usar o método `open()` para conectar-se ao banco de dados, a operação de `reencrypt()` será executada de forma síncrona. Quando a operação é concluída, a execução prossegue com a próxima linha do código.

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey
conn.reencrypt(newKey);
```

Por outro lado, se a conexão do banco de dados for aberta com o método `openAsync()`, a operação de `reencrypt()` será assíncrona. Chamar `reencrypt()` inicia o processo de recriptografia. Quando a operação é concluída, o objeto SQLConnection despacha um evento `reencrypt`. Use um ouvinte de evento para determinar quando a recriptografia será concluída:

```
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
}
```

A operação `reencrypt()` será executada na sua própria transação. Se a operação for interrompida ou falhar (por exemplo, se o aplicativo for fechado antes da conclusão da operação), a transação será revertida. Nesse caso, a chave de criptografia ainda será a chave de criptografia do banco de dados.

O método `reencrypt()` não pode ser usado para remover a criptografia de um banco de dados. Enviar um valor `null` ou chave de criptografia que não seja um ByteArray de 16 bytes para o método `reencrypt()` resulta em erro.

Considerações para usar criptografia com um banco de dados

Adobe AIR 1.5 e posterior

A seção "Uso de um banco de dados criptografado" na página 783 apresenta vários casos em que você pode usar um banco de dados criptografado. É óbvio que as situações de uso dos aplicativos diferentes (incluindo essas e outras situações) têm requisitos de privacidade diferentes. A forma que você arquiteta o uso de criptografia no seu aplicativo tem importante papel no controle do grau de privacidade dos dados de um banco de dados. Por exemplo, se você estiver usando um banco de dados criptografado para manter os dados pessoais privados, mesmo de usuários do mesmo computador, o banco de dados de cada usuário deverá ter a própria chave de criptografia. Para obter segurança máxima, seu aplicativo pode gerar a chave de uma senha digitada por um usuário. Basear a chave de criptografia em uma senha garante que os dados se mantenham preservados, mesmo se qualquer outra pessoa entrar na conta do usuário no computador. Por outro lado, suponha que você deseje que um arquivo do banco de dados seja legível para qualquer usuário do seu aplicativo, mas não para os demais aplicativos. Nesse caso, todas as cópias do aplicativo instaladas precisam de acesso a uma chave de criptografia compartilhada.

Você pode projetar seu aplicativo e, em particular, a técnica usada para gerar a chave de criptografia, de acordo com o nível de privacidade que você deseja para os dados do aplicativo. A lista a seguir fornece sugestões de design para vários níveis de privacidade de dados.

- Para tornar um banco de dados acessível a qualquer usuário que tenha acesso ao aplicativo em qualquer computador, use uma única chave disponível em todas as instâncias do aplicativo. Por exemplo, na primeira vez que um aplicativo for executado, ele poderá baixar a chave de criptografia compartilhada de um servidor usando um protocolo seguro, como SSL. Em seguida, ele poderá salvar a chave no armazenamento local criptografado para uso futuro. Como alternativa, criptografe os dados por usuário no computador e sincronize os dados com um armazenamento de dados remoto, como servidor, para tornar os dados portáteis.

- Para tornar um banco de dados acessível para um único usuário em qualquer computador, gere a chave de criptografia a partir de um segredo do usuário (como uma senha). Especificamente, não use nenhum valor associado a um computador específico (como um valor armazenado no armazenamento local criptografado) para gerar a chave. Como alternativa, criptografe os dados por usuário no computador e sincronize os dados com um armazenamento de dados remoto, como servidor, para tornar os dados portáteis.
- Para tornar um banco de dados acessível somente para um único indivíduo em um único computador, gere a chave de uma senha e um salt gerado. Para obter um exemplo dessa técnica, consulte "Exemplo: Geração e uso de uma chave de criptografia" na página 788.

A seguir há algumas considerações adicionais sobre segurança que são importantes lembrar ao projetar um aplicativo para usar um banco de dados criptografado.

- Um sistema só é seguro como seu link mais fraco. Se você estiver usando uma senha gerada pelo usuário para gerar uma chave de criptografia, considere impor o restrições de complexidade e tamanho mínimo às senhas. Uma senha curta que use somente caracteres básicos pode ser adivinhada rapidamente.
- O código-fonte de um aplicativo do AIR é armazenado no computador de um usuário em texto simples (para conteúdo HTML) ou em um formato binário facilmente descompilável (para conteúdo SWF). Como o código-fonte é acessível, dois pontos a lembrar são:
  - Nunca codifique uma chave de criptografia no seu código-fonte
  - Sempre suponha que a técnica usada para gerar uma chave de criptografia (como gerador aleatório de caracteres ou um algoritmo de hash específico) possa ser facilmente descoberta por um invasor
- A criptografia do banco de dados AIR usa o modo AES (padrão de criptografia avançado) com CCM (contador com CBC-MAC). Essa cifra de criptografia requer a combinação de uma chave inserida pelo usuário com um valor salt para ser segura. Para obter um exemplo, consulte "Exemplo: Geração e uso de uma chave de criptografia" na página 788.
- Quando você opta por criptografar um banco de dados, todos os arquivos de discos usados pelo mecanismo de banco de dados juntamente com esse banco de dados são criptografados. No entanto, o mecanismo do banco de dados retém alguns dados temporariamente em um cache na memória para aprimorar o desempenho de tempos de leitura e gravação em transações. Todos os dados residentes na memória são criptografados. Se um invasor puder acessar a memória usada por um aplicativo do AIR, por exemplo usando um depurador, os dados do banco de dados aberto atualmente e não criptografado ficarão disponíveis.

Exemplo: Geração e uso de uma chave de criptografia

Adobe AIR 1.5 e posterior

Esse aplicativo de exemplo demonstra uma técnica para gerar uma chave de criptografia. Esse aplicativo foi projetado para fornecer o nível mais alto de privacidade e segurança para dados do usuário. Um aspecto importante da segurança de dados privados é exigir que o usuário digite uma senha sempre que o aplicativo se conectar ao banco de dados. Consequentemente, conforme mostrado no exemplo, um aplicativo que requer este nível de privacidade nunca deverá armazenar diretamente a chave de criptografia do banco de dados.

O aplicativo consiste em duas partes: uma classe ActionScript que gera uma chave de criptografia (a classe EncryptionKeyGenerator) e uma interface do usuário básica que demonstra como usar a classe. Para obter o código-fonte completo, consulte "Exemplo completo de código para gerar e usar uma chave de criptografia" na página 790.

Usar a classe EncryptionKeyGenerator para obter uma cjave de criptografia segura

Adobe AIR 1.5 e posterior

Não é necessário compreender os detalhes de como a classe EncryptionKeyGenerator funciona para usá-la no seu aplicativo. Se estiver interessado nos detalhes de como a classe gera uma chave de criptografia para um banco de dados, consulte "Noções básicas da classe EncryptionKeyGenerator" na página 795.

Siga essas etapas para usar a classe EncryptionKeyGenerator no seu aplicativo:

1. Baixe a classe EncryptionKeyGenerator como código-fonte ou um SWC compilado. A classe EncryptionKeyGenerator está incluída no projeto biblioteca principal (as3corelib) do ActionScript 3.0 de código-fonte aberto. Você pode baixar o pacote as3corelib incluindo o código-fonte e a documentação. Você também pode baixar os arquivos SWC ou de código-fonte na página do projeto.
2. Coloque o código-fonte da classe EncryptionKeyGenerator (ou o SWC as3corelib) em um local onde o código-fonte do seu aplicativo possa encontrá-lo.
3. No código-fonte do aplicativo, adicione uma instrução `import` para a classe EncryptionKeyGenerator.

   ```
   import com.adobe.air.crypto.EncryptionKeyGenerator;
   ```

4. Antes do ponto onde o código cria o banco de dados ou abre uma conexão para ele, adicione o código para criar uma ocorrência EncryptionKeyGenerator chamando o construtor `EncryptionKeyGenerator()`.

   ```
   var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
   ```

5. Obtenha uma senha do usuário:

   ```
   var password:String = passwordInput.text;

   if (!keyGenerator.validateStrongPassword(password))
   {
       // display an error message
       return;
   }
   ```

   A ocorrência EncryptionKeyGenerator usa essa senha como base da chave de criptografia (mostrado na próxima etapa). A ocorrência EncryptionKeyGenerator testa a senha em relação a alguns requisitos de validação de senha forte. Em caso de falha na validação, ocorre um erro. Como o código de exemplo mostra, você pode verificar a senha antecipadamente chamando o método `validateStrongPassword()` do objeto EncryptionKeyGenerator. Dessa forma, você pode determinar se a senha atende aos requisitos mínimos de uma senha forte e evita que haja erro.

6. Gere a chave de criptografia da senha:

   ```
   var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
   ```

   O método `getEncryptionKey()` gera e retorna a chave de criptografia (um ByteArray de 16 bytes). Em seguida, você pode usar a chave de criptografia para criar seu novo banco de dados criptografado ou abrir um já existente.

   O método `getEncryptionKey()` tem um parâmetro obrigatório, que é a senha obtida na etapa 5.

   Nota: *Para manter o maior nível de segurança e privacidade dos dados, o aplicativo deve requerer que o usuário digite uma senha sempre que se conectar ao banco de dados. Não armazene diretamente a senha do usuário ou a chave de criptografia do banco de dados. Isso expõe a riscos de segurança. Em vez disso, conforme demonstrado nesse exemplo, o aplicativo deve usar a mesma técnica para derivar a chade de criptografia da senha, ao criar o banco de dados criptografado e quando se conectar a ele mais tarde.*

O método `getEncryptionKey()` também aceita um segundo parâmetro (opcional), o parâmetro `overrideSaltELSKey`. O EncryptionKeyGenerator cria um valor aleatório (conhecido como *salt*) que é usado como parte da chave de criptografia. Para ser capaz de criar novamente a chave de criptografia, o valor salt é armazenado no armazenamento local criptografado (ELS) do seu aplicativo do AIR. Por padrão, a classe EncryptionKeyGenerator usa uma String específica como chave ELS. Embora improvável, é possível que a chave possa entrar em conflito com outra chave que o aplicativo usa. Em vez de usar a chave padrão, talvez você deseje especificar sua própria chave ELS. Nesse caso, especifique uma chave personalizada, passando-a como segundo parâmetro `getEncryptionKey()`, conforme mostrado aqui:

```
var customKey:String = "My custom ELS salt key";
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password, customKey);
```

7 Criar ou abrir o banco de dados

Com a chave de criptografia retornada pelo método `getEncryptionKey()`, o código pode criar um novo banco de dados criptografado ou tentar abrir o banco de dados criptografado existente. Em ambos os casos, você usa o método `open()` ou `openAsync()` da classe SQLConnection, conforme descrito em "Criação de um banco de dados criptografado" na página 784 e "Conexão com um banco de dados criptografado" na página 785.

Nesse exemplo, o aplicativo foi projetado para abrir o banco de dados em modo de execução assíncrona. O código configura os ouvintes do evento apropriados e chama o método `openAsync()`, enviando a chave de criptografia como argumento final.

```
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, openError);

conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
```

Nos métodos do ouvinte, o código remove os registros do ouvinte do evento. Em seguida, ele exibe uma mensagem de status indicando se o banco de dados foi criado, aberto ou se ocorreu um erro. A parte mais notável desses manipuladores de eventos está no método `openError()`. Nesse método, a instrução `if` verifica se o banco de dados existe (o que significa que o código está tentando se conectar a um banco de dados existente) e se a ID do erro corresponde à constante `EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID`. Se ambas as condições forem verdadeiras, isso significa provavelmente que a senha inserida pelo usuário está incorreta. (Isso também pode significar que o arquivo especificado não é de maneira nenhuma um arquivo de banco de dados.) A seguir temos o código que verifica a ID do erro:

```
if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
{
    statusMsg.text = "Incorrect password!";
}
else
{
    statusMsg.text = "Error creating or opening database.";
}
```

Para obter o código completo dos ouvintes de eventos de exemplo, consulte "Exemplo completo de código para gerar e usar uma chave de criptografia" na página 790.

Exemplo completo de código para gerar e usar uma chave de criptografia

Adobe AIR 1.5 e posterior

A seguir há um exemplo completo de um código para a aplicação de "Geração e uso da chave de criptografia". O código consiste em duas partes.

O exemplo usa a classe EncryptionKeyGenerator para criar uma chave de criptografia a partir de uma senha. A classe EncryptionKeyGenerator está incluída no projeto biblioteca principal (as3corelib) do ActionScript 3.0 de código-fonte aberto. Você pode baixar o pacote as3corelib incluindo o código-fonte e a documentação. Você também pode baixar os arquivos SWC ou de código-fonte na página do projeto.

Exemplo do Flex

O arquivo do aplicativo MXML contém o código-fonte de um aplicativo simples que cria ou abre uma conexão para um banco de dados criptografado:

```
<?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>
```

Exemplo do Flash Professional

O arquivo FLA do aplicativo contém o código fonte de um aplicativo simples que cria ou abre uma conexão para um banco de dados criptografado. O arquivo FLA tem quatro componentes colocados no palco:

Nome da ocorrência	Tipo do componente	Descrição
`instruções`	Label	Contém as instruções fornecidas ao usuário
`passwordInput`	TextInput	Campo de entrada onde o usuário digita a senha
`openButton`	Button	Botão em que o usuário clica após digitar a senha
`statusMsg`	Label	Exibe mensagens de status (êxito ou falha)

O código do aplicativo é definido em um quadro-chave no quadro 1 da linha do tempo principal. Este é o código do aplicativo:

```
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.";
    }
}
```

Noções básicas da classe EncryptionKeyGenerator

Adobe AIR 1.5 e posterior

Não é necessário compreender o funcionamento interno da classe EncryptionKeyGenerator para usá-la com o fim de criar uma chave de criptografia segura para o banco de dados do aplicativo. O processo para usar a classe está explicado em "Usar a classe EncryptionKeyGenerator para obter uma cjave de criptografia segura" na página 789. No entanto, pode ser conveniente entender as técnicas que a classe utiliza. Por exemplo, pode ser conveniente adaptar a classe ou incorporar algumas de suas técnicas em situações nas quais um nível diferente de privacidade de dados é desejado.

A classe EncryptionKeyGenerator está incluída no projeto biblioteca principal (as3corelib) do ActionScript 3.0 de código-fonte aberto. Você pode fazer o download dopacote as3corelib, incluindo o código-fonte e a documentação. Também é possível exibir o código-fonte no site do projeto ou baixá-lo para acompanhar juntamente com as explicações.

Quando o código cria uma ocorrência EncryptionKeyGenerator e chama o respectivo método `getEncryptionKey()`, várias etapas são realizadas para garantir que apenas o usuário com permissão possa acessar os dados. O processo é o mesmo para gerar uma chave de criptografia de uma senha inserida pelo usuário antes de o banco de dados ter sido criado, bem como criar novamente a chave de criptografia para abrir o banco de dados.

Obter e validar uma senha forte

Adobe AIR 1.5 e posterior

Quando o código chama o método `getEncryptionKey()`, ele passa uma senha como um parâmetro. A senha é usada como base para a chave de criptografia. Usando informações que somente o usuário conheça, esse design garante que somente o usuário que saiba a senha possa acessar os dados do banco de dados. Mesmo se um invasor acessar a conta do usuário no computador, não poderá entrar no banco de dados sem saber a senha. Para fornecer máxima proteção, o aplicativo nunca armazena a senha.

O código de um aplicativo cria uma instância EncryptionKeyGenerator e chama o método `getEncryptionKey()`, passando uma senha inserida pelo usuário como argumento (a variável `password` neste exemplo):

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

A primeira etapa que a classe EncryptionKeyGenerator executa quando o método `getEncryptionKey()` é chamado, é verificar a senha digitada pelo usuário para garantir que ela atende aos requisitos de segurança da senha. A classe EncryptionKeyGenerator exige que a senha tenha de 8 a 32 caracteres. A senha deve conter letras maiúsculas e minúsculas e pelo menos um número ou caractere simbólico.

A expressão regular que marca esse padrão é definida como uma constante chamada de `STRONG_PASSWORD_PATTERN`:

```
private static const STRONG_PASSWORD_PATTERN:RegExp =
/(?=^.{8,32}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/;
```

O código que verifica a senha está no método `validateStrongPassword()` da classe EncryptionKeyGenerator. O código é o seguinte:

```
public function vaidateStrongPassword(password:String):Boolean
{
    if (password == null || password.length <= 0)
    {
        return false;
    }

    return STRONG_PASSWORD_PATTERN.test(password))
}
```

Internamente, o método `getEncryptionKey()` chama o método `validateStrongPassword()` da classe EncryptionKeyGenerator e, se a senha não é válida, lança uma exceção. O método `validateStrongPassword()` é um método público para que o código do aplicativo possa verificar a senha sem chamar o método `getEncryptionKey()`, para evitar um erro.

Expandir a senha para 256 bits

Adobe AIR 1.5 e posterior

Mais tarde no processo, a senha deverá ter 256 bits. Em vez de exigir que cada usuário insira uma senha com exatamente 256 bits (32 caracteres), o código cria uma senha maior, repetindo os caracteres da senha.

O método `getEncryptionKey()` chama o método `concatenatePassword()` para executar a tarefa de criar a senha longa.

```
var concatenatedPassword:String = concatenatePassword(password);
```

A seguir, temos o código do método `concatenatePassword()`:

```
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;
}
```

Se a senha tem menos de 256 bits, o código concatena a senha com ele mesmo para torná-la 256 bits. Se o comprimento não funcionar exatamente, a última repetição será reduzida para obter exatamente 256 bits.

Gerar ou recuperar um valor de salt de 256 bits

Adobe AIR 1.5 e posterior

A próxima etapa é obter um valor de salt de 256 bits que, posteriormente, será combinado com a senha. Um *salt* é um valor aleatório adicionado ou combinado com um valor inserido pelo usuário para formar uma senha. Usar um salt com uma senha garante que mesmo se um usuário escolher uma palavra real ou termo comum como senha, a combinação senha mais salt que o sistema usa será um valor aleatório. Isso, de maneira não aleatória, ajuda a proteger contra um ataque ao dicionário, em que um invasor usa uma lista de palavras para tentar adivinhar uma senha. Além disso, gerando o valor salt e armazenando-o no armazenamento local criptografado, ele é associado à conta do usuário no computador em que o arquivo do banco de dados está localizado.

Se o aplicativo estiver chamando o método `getEncryptionKey()` pela primeira vez, o código criará um valor salt aleatório de 256 bits. Caso contrário, o código carregará o valor de salt do armazenamento local criptografado.

O salt é armazenado em uma variável chamada de `salt`. O código determina se já foi criado um salt, ao tentar carregar o salt do armazenamento local criptografado:

```
var salt:ByteArray = EncryptedLocalStore.getItem(saltKey);
if (salt == null)
{
    salt = makeSalt();
    EncryptedLocalStore.setItem(saltKey, salt);
}
```

Se o código estiver criando um novo valor salt, o método `makeSalt()` gera um valor aleatório de 256 bits. Como o valor é eventualmente armazenado no armazenamento local criptografado, ele é gerado como um objeto ByteArray. O método `makeSalt()` usa o método `Math.random()` para gerar aleatoriamente o valor. O método `Math.random()` não pode gerar 256 bits de uma vez. Em vez disso, o código usa um loop para chamar o `Math.random()` oito vezes. A cada vez, um valor uint aleatório entre 0 e 4294967295 (o valor uint máximo) é gerado. Um valor uint é usado por conveniência, pois um uint usa exatamente 32 bits. Quando oito valores uint são gravados no ByteArray, é gerado um valor de 256 bits. Este é o código para o método `makeSalt()`:

```
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;
}
```

Quando o código está salvando o salt no armazenamento local criptografado (ELS) ou recuperando o salt do ELS, ele precisa da chave String com a qual o salt é salvo. Sem saber a chave, o valor salt não pode ser recuperado. Nesse caso, a chave de criptografia não pode ser recriada todas as vezes para abrir novamente o banco de dados. Por padrão, o EncryptionKeyGenerator usa uma chave ELS predefinida que é definida na constante `SALT_ELS_KEY`. Em vez de usar a chave padrão, o código do aplicativo também pode especificar uma chave ELS para usar na chamada do método `getEncryptionKey()`. A Chave ELS padrão ou a chave ELS salt especificada pelo aplicativo é armazenada em uma variável chamada de `saltKey`. Essa variável é usada nas chamadas a `EncryptedLocalStore.setItem()` e `EncryptedLocalStore.getItem()`, como mostrado antes.

Combinar a senha de 256 bits e o salt usando o operador XOR

Adobe AIR 1.5 e posterior

O código agora tem uma senha de 256 bits e um valor salt de 256 bits. Em seguida, ele usa uma operação XOR em nível de bits para combinar o salt e a senha concatenada em um único valor. Efetivamente, essa técnica cria um senha de 256 bits que consiste em caracteres do intervalo inteiro de caracteres possíveis. Esse princípio é verdadeiro, embora muito provavelmente a entrada de senha real consista principalmente em caracteres alfanuméricos. Essa não aleatoriedade elevada oferece o benefício de tornar o conjunto de senhas possíveis maior, sem exigir que o usuário digite uma senha longa e complexa.

O resultado da operação XOR é armazenado na variável `unhashedKey`. O processo real de execução de um XOR em nível de bits nos dois valores ocorre no método `xorBytes()`:

```
var unhashedKey:ByteArray = xorBytes(concatenatedPassword, salt);
```

O operador XOR em nível de bits (`^`) retira dois valores uint e retorna um valor uint. (Um valor uint contém 32 bits.) Os valores de entrada enviados como argumentos para o método `xorBytes()` são uma String (a senha) e um ByteArray (o salt). Consequentemente, o código usa um loop para extrair 32 bits de uma vez de cada entrada para combinar usando o operador XOR.

```
private function xorBytes(passwordString:String, salt:ByteArray):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 4)
    {
        // ...
    }

    return result;
}
```

No loop, os primeiros 32 bits (4 bytes) são extraídos do parâmetro `passwordString`. Esses bits são extraídos e convertidos em um uint (`o1`) em um processo de duas partes. Primeiro, o método `charCodeAt()` obtém cada valor numérico do caractere. Depois, esse valor é desviado para a posição apropriada na unidade usando o operador de desvio à esquerda em nível de bits (`<<`) e o valor desviado é adicionado ao `o1`. Por exemplo, o primeiro caractere (`i`) se torna os primeiros 8 bits usando o operador de desvio à esquerda em nível de bits (`<<`) para desviar os bits deixados por 24 bits e atribuindo esse valor ao `o1`. O segundo caractere `(i + 1)` se torna os segundos 8 bits desviando seu valor 16 bits à esquerda e adicionando o resultado a `o1`. Os valores do terceiro e do quarto caracteres são adicionados da mesma forma.

```
        // ...

        // 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);

        // ...
```

A variável `o1` agora contém 32 bits do parâmetro `passwordString`. Em seguida, 32 bits são extraídos do parâmetro `salt`, chamando seu método `readUnsignedInt()`. Os 32 bits são armazenados em uma variável uint `o2`.

```
        // ...

        salt.position = i;
        var o2:uint = salt.readUnsignedInt();

        // ...
```

Finalmente, os dois valores de 32 bits (uint) são combinados usando o operador XOR, e o resultado é gravado em um ByteArray chamado de `result`.

```
        // ...

        var xor:uint = o1 ^ o2;
        result.writeUnsignedInt(xor);
        // ...
```

Quando o loop é concluído, o ByteArray contendo o resultado XOR é retornado.

```
        // ...
    }

    return result;
}
```

Hash da chave

Adobe AIR 1.5 e posterior

Após a senha concatenada e o salt serem combinados, a próxima etapa é oferecer proteção adicional a esse valor, fazendo hash dele por meio do algoritmo de hash SHA-256. O processo de hash dificulta para um invasor realizar a engenharia reversa no valor.

Nesse ponto, o código tem um ByteArray chamado de `unhashedKey`, que contém a senha concatenada combinada com o salt. O projeto de biblioteca central (as3corelib) do ActionScript 3.0 inclui uma classe SHA256 no pacote com.adobe.crypto. O método `SHA256.hashBytes()`, que realiza um hash de SHA-256 em um ByteArray e retorna uma String contendo o resultado do hash de 256 bits como um número decimal. A classe EncryptionKeyGenerator usa a classe SHA256 para aplicar hash à chave:

```
var hashedKey:String = SHA256.hashBytes(unhashedKey);
```

Extrair a chave de criptografia do hash

Adobe AIR 1.5 e posterior

A chave de criptografia deve ser um ByteArray com exatamente 16 bytes (128 bits). O resultado do algoritmo de hash SHA-256 tem sempre 256 bits. Consequentemente, a etapa final é selecionar 128 bits do resultado com hash para usar como chave de criptografia real.

Na classe EncryptionKeyGenerator, o código reduz a chave para 128 bits chamando o método `generateEncryptionKey()`. Em seguida, ele retorna esse resultado do método como o resultado do método `getEncryptionKey()`:

```
var encryptionKey:ByteArray = generateEncryptionKey(hashedKey);
return encryptionKey;
```

É necessário usar os primeiros 128 bits como a chave de criptografia. Você pode selecionar um intervalo de bits iniciando em algum ponto arbitrário, pode selecionar todos os outros bits ou usar alguma outra maneira de selecionar bits. O importante é que o código selecione 128 bits diferentes e que os mesmos 128 bits sejam usados a cada vez.

Nesse caso, o método `generateEncryptionKey()` usa o intervalo de bits que começa no 18º byte como a chave de criptografia. Como mencionado antes, a classe SHA256 retorna uma String contendo um hash de 256 bits como número hexadecimal. Um único bloco de 128 bits não tem muitos bytes a acrescentar a um ByteArray de uma vez. Consequentemente, o código usa um loop `for` para extrair caracteres da String hexadecimal, convertendo-os em valores numéricos reais e adicionando-os ao ByteArray. A String de resultado SHA-256 tem 64 caracteres. Um intervalo de 128 bits equivale a 32 caracteres na String, e cada caractere representa 4 bits. O menor incremento de dados que você pode adicionar a um ByteArray é um byte (8 bits), o que é equivalente a dois caracteres na String `hash`. Consequentemente, o loop conta de 0 a 31 (32 caracteres) em incrementos de 2 caracteres.

Dentro do loop, o código determina primeiro a posição inicial do par de caracteres atual. Como o intervalo desejado começa na posição de indexação 17 (o 18º byte) do caractere, a variável `position` é atribuída ao valor iterador da corrente (`i`) mais 17. O código usa o método `substr()` do objeto da String para extrair os dois caracteres na posição atual. Esses caracteres são armazenados na variável `hex`. Em seguida, o código usa o método `parseInt()` para converter a String `hex` a um valor inteiro decimal. Ele armazena esse valor na variável `byte`. Finalmente, o código adiciona o valor de `byte` ao ByteArray `result` usando o método `writeByte()`. Quando o loop é concluído, o ByteArray `result` contém 16 bytes e está pronto para ser usado como chave de criptografia do banco de dados.

```
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;
}
```

Estratégias para trabalhar com bancos de dados SQL

Adobe AIR 1.0 e posterior

Existem várias maneiras pelas quais um aplicativo pode acessar e trabalhar com um banco de dados SQL local. O design do aplicativo pode variar em termos de como seu código é organizado, a sequência e o intervalo de execução das operações, e assim por diante. As técnicas que você escolhe podem afetar a facilidade com que o aplicativo é desenvolvido. Elas podem afetar a facilidade com que você modificará o aplicativo em futuras atualizações. Elas também podem afetar o desempenho do aplicativo do ponto de vista dos usuários.

Distribuição de um banco de dados previamente preenchido

Adobe AIR 1.0 e posterior

Quando você usa um banco de dados SQL local do AIR no seu aplicativo, este espera um banco de dados com uma certa estrutura de tabelas, colunas e assim por diante. Alguns aplicativos também esperam que certos dados estejam previamente preenchidos no arquivo de banco de dados. Uma forma de assegurar que o banco de dados tenha a estrutura apropriada é criar um banco de dados no código do aplicativo. Quando o aplicativo é carregado, ele verifica se o arquivo de banco de dados correspondente existe em um determinado local. Se o arquivo não existir, o aplicativo executará um conjunto de comandos para criar o arquivo de banco de dados, criar a estrutura do banco de dados e preencher as tabelas com os dados iniciais.

O código que cria o banco de dados e suas tabelas geralmente é complexo. Com frequência, ele só é usado uma vez durante o período em que o aplicativo fica instalado, mas ainda assim aumenta o tamanho e a complexidade do aplicativo. Como alternativa à criação do banco de dados, da estrutura e dos dados de modo programático, você pode distribuir um banco de dados previamente preenchido com o aplicativo. Para distribuir um banco de dados predefinido, inclua o arquivo correspondente no pacote AIR do aplicativo.

Como todos os arquivos que são incluídos em um pacote AIR, um arquivo de banco de dados compactado é instalado no diretório do aplicativo (o diretório representado pela propriedade `File.applicationDirectory`). No entanto, os arquivos desse diretório são somente leitura. Use o arquivo do pacote AIR como um banco de dados "modelo". Na primeira vez que um usuário executar o aplicativo, copie o arquivo de banco de dados original para o "Apontar para o diretório de armazenamento do aplicativo" na página 694 do usuário (ou outro local) e use esse banco de dados no aplicativo.

Melhores práticas de trabalho com bancos de dados SQL locais

Adobe AIR 1.0 e posterior

A lista a seguir é um conjunto de técnicas sugeridas que você pode usar para melhorar o desempenho, a segurança e a facilidade de manutenção dos seus aplicativos quando trabalhar com bancos de dados SQL locais.

Crie conexões de banco de dados previamente

Adobe AIR 1.0 e posterior

Mesmo que o seu aplicativo não execute nenhuma instrução quando inicialmente carregado, instancie um objeto SQLConnection e chame o método `open()` ou `openAsync()` correspondente com antecedência (por exemplo, após a inicialização do aplicativo) para evitar atrasos na execução de instruções. Consulte "Conexão com um banco de dados" na página 750.

Reutilize conexões de banco de dados

Adobe AIR 1.0 e posterior

Se você acessa um certo banco de dados ao longo da execução do seu aplicativo, mantenha uma referência à ocorrência de SQLConnection e a reutilize no aplicativo inteiro em vez de fechar e reabrir a conexão. Consulte "Conexão com um banco de dados" na página 750.

Dê preferência ao modo de execução assíncrona

Adobe AIR 1.0 e posterior

Quando criamos um código de acesso a dados, pode ser tentador executar operações de maneira síncrona em vez de assíncrona, porque usar operações síncronas geralmente requer um código mais curto e menos complexo. Porém, conforme descrito na seção "Uso de operações de banco de dados síncronas e assíncronas" na página 777, as operações síncronas podem ter um impacto no desempenho que é óbvio para os usuários e prejudicial para a experiência deles no aplicativo. O tempo usado para uma única operação varia conforme a operação e, principalmente, o volume de dados envolvido. Por exemplo, uma instrução SQL `INSERT` que só adiciona uma linha ao banco de dados demora menos tempo do que uma instrução `SELECT` que recupera milhares de linhas de dados. No entanto, quando você usa a execução síncrona para realizar várias operações, as operações normalmente são encadeadas. Mesmo que o tempo que cada operação leve seja muito curto, o aplicativo fica congelado até a conclusão de todas as operações síncronas. Por isso, o tempo acumulado de várias operações encadeadas pode ser suficiente para paralisar o aplicativo.

Use operações assíncronas como uma abordagem padrão, principalmente no caso de operações que envolvem um grande número de linhas. Há uma técnica para dividir o processamento de grandes conjuntos de resultados de instruções `SELECT`, descrita em "Recuperação dos resultados de SELECT em partes" na página 766. No entanto, ela só pode ser usada no modo de execução assíncrona. Utilize operações síncronas apenas quando não conseguir obter uma certa funcionalidade usando programação assíncrona, quando tiver considerado as vantagens e desvantagens para os usuários do aplicativo em termos de desempenho e quando testar o aplicativo e perceber que o desempenho foi afetado. O uso da execução assíncrona pode envolver uma codificação mais complexa. Porém, lembre-se de que você só precisa criar o código uma vez, mas que os usuários do aplicativo têm de usá-lo repetidas vezes, de forma ágil ou lentamente.

Em muitos casos, usando uma ocorrência de SQLStatement à parte para cada instrução SQL a ser executada, é possível enfileirar várias operações SQL de uma só vez, o que torna o código assíncrono parecido com o código síncrono quanto à maneira como ele é criado. Para obter mais informações, consulte "Noções básicas sobre o modelo de execução assíncrona" na página 781.

Use instruções SQL separadas e não altere a propriedade text de SQLStatement

Adobe AIR 1.0 e posterior

Para qualquer instrução SQL executada mais de uma vez em um aplicativo, crie uma ocorrência de SQLStatement à parte para cada instrução SQL. Use essa ocorrência de SQLStatement sempre que esse comando SQL for executado. Por exemplo, suponha que você está criando um aplicativo que inclui quatro operações SQL diferentes executadas várias vezes. Nesse caso, crie quatro ocorrências separadas de SQLStatement e chame o método `execute()` de cada instrução para executá-la. Evite a alternativa de usar uma única ocorrência de SQLStatement para todas as instruções SQL, sempre redefinindo sua propriedade `text` antes de executar a instrução.

Use parâmetros de instrução

Adobe AIR 1.0 e posterior

Use parâmetros SQLStatement — nunca concatene a entrada do usuário no texto da instrução. O uso de parâmetros torna o seu aplicativo mais seguro porque impede a possibilidade de ataques de injeção SQL. Isso permite usar objetos em consultas (e não apenas valores literais SQL). Também torna a execução das instruções mais eficiente, porque elas podem ser reutilizadas sem que você precise recompilá-las todas as vezes que forem executadas. Consulte "Uso de parâmetros em instruções" na página 754 para obter mais informações.

Capítulo 41: Trabalhar com matrizes de bytes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe ByteArray permite que você leia de e escreva para um fluxo de dados binário, que é essencialmente uma matriz de bytes. Essa classe fornece uma maneira de acessar dados no nível mais elementar. Como os dados do computador consistem em bytes, ou grupos de 8 bits, a capacidade de ler dados em bytes significa que você pode acessar dados para os quais classes e métodos de acesso não existem. A classe ByteArray permite que você analise qualquer fluxo de dados, de um bitmap a um fluxo de dados que viaje pela rede, no nível de byte.

O método `writeObject()` permite que você escreva um objeto em AMF serializado para um ByteArray, enquanto o método `readObject()` permite que você leia um objeto serializado de um ByteArray para uma variável do tipo de dados original. Você pode serializar qualquer objeto, exceto objetos de exibição, que são aqueles que podem ser colocados na lista de exibição. Você também pode atribuir objetos serializados de volta às instâncias de classe personalizada se a classe personalizada estiver disponível para o tempo de execução. Após converter um objeto para AMF, você pode transferi-lo de modo eficiente por uma conexão de rede ou salvá-lo em um arquivo.

O aplicativo de exemplo do Adobe® AIR® descrito aqui lê um arquivo .zip como exemplo de processar um fluxo de bytes, extrair uma lista dos arquivos contidos no arquivo .zip e escrevê-los na área de trabalho.

Mais tópicos da Ajuda

flash.utils.ByteArray

flash.utils.IExternalizable

Especificação do Action Message Format

Ler e escrever um ByteArray

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe ByteArray é parte do pacote flash.utils. Para criar um objeto ByteArray no ActionScript 3.0, importe a classe ByteArray e invoque o construtor, conforme exibido no seguinte exemplo:

```
import flash.utils.ByteArray;
var stream:ByteArray = new ByteArray();
```

Métodos ByteArray

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Qualquer fluxo de dados significativo é organizado em um formato que você pode analisar para encontrar as informações desejadas. Um registro em um simples arquivo de um funcionário, por exemplo, provavelmente incluiria um número de ID, um nome, um endereço, um número de telefone e assim por diante. Um arquivo de áudio MP3 contém uma marca ID3 que identifica o título, autor, álbum, data de publicação e gênero do arquivo que está sendo obtido por download. O formato permite que você saiba a ordem na qual esperar os dados no fluxo de dados. Ele permite que você leia o fluxo de bytes de modo inteligente.

A classe ByteArray inclui vários métodos que facilitam ler de e escrever para um fluxo de dados. Alguns desses métodos incluem `readBytes()` e `writeBytes()`, `readInt()` e `writeInt()`, `readFloat()` e `writeFloat()`, `readObject()` e `writeObject()` e `readUTFBytes()` e `writeUTFBytes()`. Esses métodos permitem que você leia dados do fluxo de dados em variáveis de tipos de dados específicos e escreva de tipos de dados específicos diretamente para o fluxo de dados binário.

Por exemplo, o código a seguir lê uma matriz simples de sequências de caracteres e números de ponto flutuante e escreve cada elemento em um ByteArray. A organização da matriz permite que o código chame os métodos ByteArray apropriados (`writeUTFBytes()` e `writeFloat()`) para escrever os dados. O padrão de dados repetitivo possibilita ler a matriz com um loop.

```
 // 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
```

A propriedade position

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A propriedade position armazena a posição atual do ponteiro que indexa ByteArray durante a leitura ou escrita. O valor inicial da propriedade position é 0 (zero), como exibido no seguinte código:

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
```

Quando você lê de ou escreve em um ByteArray, o método usado atualiza a propriedade position para apontar para o local imediatamente seguindo o último byte lido ou escrito. Por exemplo, o código a seguir escreve uma sequência de caracteres em um ByteArray e, posteriormente, a propriedade position aponta para o byte imediatamente seguindo a sequência de caracteres no 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
```

Da mesma forma, uma operação de leitura incrementa a propriedade position pelo número de bytes lidos.

```
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!
```

Observe que você pode definir a propriedade position para um local específico no ByteArray para ler ou escrever naquele deslocamento.

As propriedades bytesAvailable e length

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As propriedades `length` e `bytesAvailable` dizem a você quanto tempo um ByteArray possui e quantos bytes permanecem nele da posição atual até o fim. O exemplo a seguir ilustra como você pode usar essas propriedades. O exemplo escreve uma sequência de caracteres de texto para o ByteArray e, em seguida, lê um byte de cada vez do ByteArray até que ele encontre o caractere “a” ou o final (`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
}
```

A propriedade endian

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os computadores podem diferir em como armazenam números de vários bytes, isto é, números que exigem mais de um 1 byte de memória para armazená-los. Um inteiro, por exemplo, pode levar 4 bytes, ou 32 bits, de memória. Alguns computadores armazenam o byte mais significativo do número primeiro, no endereço de memória mais baixo e outros armazenam o byte menos significativo primeiro. Esse atributo de um computador, ou de uma ordem de bytes, é conhecido como sendo *big endian* (byte mais significativo primeiro) ou *little endian* (byte menos significativo primeiro). Por exemplo, o número 0x31323334 seria armazenado da seguinte forma para ordem de bytes big endian e little endian, onde a0 representa o endereço de memória mais baixo dos 4 bytes e a3 representa o mais alto:

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

A propriedade `endian` da classe ByteArray permite que você denote essa ordem de bytes para números de vários bytes que você esteja processando. Os valores aceitáveis para essa propriedade são `"bigEndian"` ou `"littleEndian"` e a classe Endian define as constantes `BIG_ENDIAN` e `LITTLE_ENDIAN` para definir a propriedade `endian` com essas sequências de caracteres.

Os métodos compress() e uncompress()

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `compress()` permite que você compacte um ByteArray de acordo com um algoritmo de compactação especificado como um parâmetro. O método `uncompress()` permite que você descompacte um ByteArray compactado de acordo com um algoritmo de compactação. Após chamar `compress()` e `uncompress()`, o comprimento da matriz de bytes é definida para o novo comprimento e a propriedade position é definida para o fim.

A classe CompressionAlgorithm define constantes que você pode usar para especificar o algoritmo de compactação. A classe ByteArray suporta os algoritmos deflate (somente no AIR), zlib e lzma. O formato dos dados comprimidos zlib está descrito em http://www.ietf.org/rfc/rfc1950.txt_br. O algoritmo lzma foi adicionado ao Flash Player 11.4 e ao AIR 3.4. Ele é descrito em http://www.7-zip.org/7z.html.

O algoritmo de compactação deflate é usado em vários formatos de compactação, tais como zlib, gzip e algumas implementações zip. O algoritmo de compactação deflate está descrito em http://www.ietf.org/rfc/rfc1951.txt_br.

O exemplo a seguir compacta um ByteArray chamado `bytes` usando o algoritmo lzma:

```
bytes.compress(CompressionAlgorithm.LZMA);
```

O exemplo a seguir descompacta um ByteArray compactado usando o algoritmo deflate:

```
bytes.uncompress(CompressionAlgorithm.LZMA);
```

Leitura e escrita de objetos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os métodos `readObject()` e `writeObject()` leem um objeto de e escrevem um objeto para um ByteArray, codificado em AMF serializado. AMF é um protocolo de mensagens proprietário criado pela Adobe e usado por várias classes do ActionScript 3.0, incluindo Netstream, NetConnection, NetStream, LocalConnection e Shared Objects.

Um marcador de tipo de um byte descreve o tipo dos dados codificados que segue. O AMF usa os 13 tipos de dados a seguir:

```
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
```

Os dados codificados seguem o marcador de tipo, a menos que o marcador represente um único valor possível, como null ou true ou false, em cujo caso nada mais é codificado.

Existem duas versões do AMF: AMF0 e AMF3. O AMF 0 suporta o envio de objetos complexos por referência e permite pontos de extremidade para restaurar as relações entre objetos. O AMF 3 melhora o AMF 0 enviando características de objetos e sequências de caracteres por referência, além de referências de objetos, e suportando novos tipos de dados introduzidos no ActionScript 3.0. A propriedade `ByteArray.objectEcoding` especifica a versão do AMF usada para codificar os dados do objeto. A classe flash.net.ObjectEncoding define constantes para especificar a versão do AMF: `ObjectEncoding.AMF0` e `ObjectEncoding.AMF3`.

O exemplo a seguir chama `writeObject()` para escrever um objeto XML para um ByteArray, que ele, em seguida compacta usando o algoritmo Deflate e escreve para o arquivo `order` na área de trabalho. O exemplo usa um rótulo para exibir a mensagem “Arquivo de ordem escrito na área de trabalho!” na janela do AIR quando concluído.

```
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();
    }
}
```

O método `readObject()` lê um objeto no AMF serializado de um ByteArray e o armazena em um objeto do tipo especificado. O exemplo a seguir lê o arquivo `order` da área de trabalho em um ByteArray (`inBytes`), o descompacta e chama `readObject()` para armazená-lo no objeto XML `orderXML`. O exemplo usa uma construção de loop `for each()` para adicionar cada nó a uma área de texto para exibição. O exemplo também exibe o valor da propriedade `objectEncoding` juntamente com um cabeçalho para o conteúdo do arquivo `order`.

```
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();
    }
}
```

Exemplo de ByteArray: leitura de um arquivo .zip

Adobe AIR 1.0 e posterior

Esse exemplo demonstra como ler um simples arquivo .zip contendo vários arquivos de diferentes tipos. Ele faz isso extraindo dados relevantes dos metadados para cada arquivo, descompactando cada arquivo em um ByteArray e escrevendo o arquivo na área de trabalho.

A estrutura geral de um arquivo .zip é baseada na especificação de PKWARE Inc., mantida em http://www.pkware.com/documents/casestudies/APPNOTE.TXT. Primeiramente, está o cabeçalho de um arquivo e os dados do arquivo para o primeiro arquivo no arquivo .zip, seguido por um cabeçalho de arquivo e par de dados de arquivo para cada arquivo adicional. (A estrutura do cabeçalho do arquivo é descrita posteriormente.) Em seguida, o arquivo .zip inclui opcionalmente um registro de descritor de dados (normalmente quando o arquivo zip de saída foi criado na memória, e não salvo em um disco). Em seguida, estão vários elementos opcionais adicionais: cabeçalho de descriptografia do arquivo, registro de dados extra do arquivo, estrutura de diretório central, final Zip64 de registro de diretório central, final Zip64 de localizador de diretório central e final de registro de diretório central.

O código nesse exemplo é escrito para analisar apenas arquivos zip que não contêm pastas e ele não espera registros de descritor de dados. Ele ignora todas as informações seguindo os dados do último arquivo.

O formato do cabeçalho do arquivo para cada arquivo é o seguinte:

assinatura do cabeçalho do arquivo	4 bytes
versão necessária	2 bytes
sinalizador de bits de propósito geral	2 bytes
método de compactação	2 bytes (8=DEFLATE; 0=UNCOMPRESSED)
última hora de modificação do arquivo	2 bytes
última data de modificação do arquivo	2 bytes

crc-32	4 bytes
tamanho compactado	4 bytes
tamanho descompactado	4 bytes
comprimento do nome do arquivo	2 bytes
comprimento de campo extra	2 bytes
nome de arquivo	variável
campo extra	variável

Seguindo o cabeçalho do arquivo estão os dados reais do arquivo, que podem ser compactados ou não, dependendo do sinalizador de método de compactação. O sinalizador é 0 (zero) se os dados do arquivo são descompactados, 8 se os dados são compactados usando o algoritmo DEFLATE ou outro valor para outros algoritmos de compactação.

A interface de usuário para esse exemplo consiste de um rótulo e uma área de texto (`taFiles`). O aplicativo escreve as seguintes informações na área de texto para cada arquivo encontrado no arquivo .zip: o nome de arquivo, o tamanho compactado e o tamanho descompactado. O documento MXML a seguir define a interface do usuário para a versão Flex do aplicativo:

```
<?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>
```

O início do programa executa as seguintes tarefas:

- Importa as classes necessárias

```
import flash.filesystem.*;
import flash.utils.ByteArray;
import flash.events.Event;
```

- Define a interface de usuário para 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);
```

- Define o ByteArray `bytes`

```
var bytes:ByteArray = new ByteArray();
```

- Define variáveis para armazenar metadados do cabeçalho do arquivo

```
// 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;
```

- Define os objetos File (`zfile`) e FileStream (`zStream`) para representar o arquivo .zip e especifica o local do arquivo .zip a partir do qual os arquivos são extraídos — um arquivo chamado "HelloAIR.zip" no diretório da área de trabalho.

```
// File variables for accessing .zip file
var zfile:File = File.desktopDirectory.resolvePath("HelloAIR.zip");
var zStream:FileStream = new FileStream();
```

No Flex, o código do programa começa no método `init()`, chamado como manipulador `creationComplete` para a marca raiz `mx:WindowedApplication`.

```
// for Flex
private function init():void
{
```

O programa começa abrindo o arquivo .zip em modo READ.

```
zStream.open(zfile, FileMode.READ);
```

Ele, em seguida, define a propriedade `endian` de `bytes` para `LITTLE_ENDIAN` para indicar que a ordem de bytes de campos numéricos possui o byte menos significativo primeiro.

```
bytes.endian = Endian.LITTLE_ENDIAN;
```

Em seguida, uma instrução `while()` começa um loop que continua até que a posição atual no fluxo do arquivo seja maior que ou igual ao tamanho do arquivo.

```
while (zStream.position < zfile.size)
{
```

A primeira instrução dentro do loop lê os primeiros 30 bytes do fluxo do arquivo no ByteArray `bytes`. Os primeiros 30 bytes formam a parte de tamanho fixo do cabeçalho do primeiro arquivo.

```
    // read fixed metadata portion of local file header
    zStream.readBytes(bytes, 0, 30);
```

Em seguida, o código lê um inteiro (`signature`) dos primeiros bytes do cabeçalho de 30 bytes. A definição do formato ZIP especifica que a assinatura para cada cabeçalho de arquivo é o valor hexadecimal `0x04034b50`; se a assinatura for diferente, significa que o código se moveu além da parte do arquivo .zip e não existem mais arquivos para extrair. Nesse caso, o código sai do loop `while` imediatamente, em vez de esperar pelo final da matriz de bytes.

```
bytes.position = 0;
signature = bytes.readInt();
// if no longer reading data files, quit
if (signature != 0x04034b50)
{
    break;
}
```

A parte seguinte do código lê o byte do cabeçalho na posição de deslocamento 8 e armazena o valor na variável `compMethod`. Esse byte contém um valor que indica o método de compactação usado para compactar esse arquivo. Vários métodos de compactação são permitidos, mas, na prática, praticamente todos os arquivos .zip usam o algoritmo de compactação DEFLATE. Se o arquivo atual é compactado com a compactação DEFLATE, `compMethod` é 8; se o arquivo é descompactado, `compMethod` é 0.

```
bytes.position = 8;
compMethod = bytes.readByte();  // store compression method (8 == Deflate)
```

Seguindo os primeiros 30 bytes está uma parte de comprimento variável do cabeçalho que contém o nome do arquivo e, possivelmente, um campo extra. A variável `offset` armazena o tamanho dessa parte. O tamanho é calculado pela adição do comprimento do nome do arquivo e o comprimento do campo extra, lido do cabeçalho nos deslocamentos 26 e 28.

```
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
```

Em seguida, o programa lê a parte de comprimento variável do cabeçalho do arquivo para o número de bytes armazenados na variável `offset`.

```
// read variable length bytes between fixed-length header and compressed file data
zStream.readBytes(bytes, 30, offset);
```

O programa lê o nome do arquivo da parte de comprimento variável do cabeçalho e o exibe na área de texto, juntamente com os tamanhos compactados (zip) e descompactados (original) do arquivo.

```
// 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';
```

O exemplo lê o resto do arquivo do fluxo de arquivos em `bytes` para o comprimento especificado pelo tamanho compactado, substituindo o cabeçalho do arquivo nos primeiros 30 bytes. O tamanho compactado é preciso, mesmo se o arquivo não está compactado, porque, nesse caso, o tamanho compactado é igual ao tamanho descompactado do arquivo.

```
    // 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);
```

Em seguida, o exemplo descompacta o arquivo compactado e chama a função `outfile()` para escrevê-la no fluxo de arquivos de saída. Ele transmite a `outfile()` o nome do arquivo e a matriz de bytes que contém os dados do arquivo.

```
        if (compMethod == 8) // if file is compressed, uncompress
        {
            bytes.uncompress(CompressionAlgorithm.DEFLATE);
        }
        outFile(fileName, bytes);   // call outFile() to write out the file
```

No exemplo mencionado anteriormente, `bytes.uncompress(CompressionAlgorithm.DEFLATE)` funcionará somente em aplicativos do AIR. Para expandir os dados não compactados do AIR e do Flash Player, utilize a função de expansão `inflate()`.

As chaves de fechamento indicam o final do loop `while` e do método `init()` e do código do aplicativo Flex, exceto para o método `outFile()`. A execução volta ao início do loop `while` e continua processando os próximos bytes no arquivo .zip — extraindo outro arquivo ou finalizando o processamento do arquivo .zip se o último arquivo tiver sido processado.

```
    } // end of while loop
} // for Flex version, end of init() method and application
```

A função `outfile()` abre um arquivo de saída no modo WRITE na área de trabalho, dando a ele o nome fornecido pelo parâmetro `filename`. Ela, em seguida, escreve os dados do arquivo do parâmetro `data` para o fluxo do arquivo de saída (`outStream`) e fecha o arquivo.

```
// 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();
}
```

Capítulo 42: Noções básicas de rede e comunicação

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao criar aplicativos no Flash Player ou no AIR, você precisa frequentemente acessar recursos fora de seu aplicativo. Por exemplo, você pode enviar uma solicitação por uma imagem para um servidor web e receber os dados da imagem como retorno. Ou, você pode enviar objetos serializados de um lado para o outro sobre uma conexão socket com um servidor de aplicativos. As APIs do Flash Player e do AIR fornecem várias classes que permitem a seus aplicativos participação nesta troca. Estas APIs suportam rede com base em IP para protocolos como UDP,TCP, HTTP, RMTP e RTMFP.

AS classes a seguir podem ser utilizadas para enviar e receber dados através de uma rede:

Classe	Formatos suportados de dados	Protocolos	Descrição
Loader	SWF, PNG, JPEG, GIF	HTTP, HTTPS	Carrega tipos de dados com suporte e converte os dados em objetos de exibição. Consulte "Carregamento dinâmico do conteúdo da exibição" na página 202.
URLLoader	Qualquer (texto, XML, binário, etc.)	HTTP, HTTPS	Carrega formatos arbitrários de dados. Seu aplicativo é responsável por interpretar os dados. Consulte "Uso da classe URLLoader" na página 842
FileReference	Qualquer elemento	HTTP	Enviar e baixar arquivos. Consulte "Uso da classe FileReference" na página 673
NetConnection	Vídeo, áudio, Formato de mensagem ActionScript (ActionScript Message Format AMF)	HTTP, HTTPS, RTMP e RTMFP	Faz conexão de vídeo, áudio e streams de objetos remotos. Consulte "Trabalho com vídeo" na página 487.
Som	Áudio	HTTP	Carrega e reproduz formatos de áudio com suporte. Consulte "Carregamento de arquivos de som externos" na página 456.
XMLSocket	XML	TCP	Troca mensagens XML como um servidor XMLSocket. Consulte "Soquetes XML" na página 829.
Socket	Qualquer elemento	TCP	Conecta a um servidor de socket TCP. Consulte "Soquetes binários de cliente" na página 824.
SecureSocket (AIR)	Qualquer elemento	TCP com SSLv3 ou TLSv1	Conecta a um servidor de socket TCP que requer segurança SSL ou TLS. Consulte "Soquetes de cliente seguros (AIR)" na página 824.
ServerSocket (AIR)	Qualquer elemento	TCP	Atua como um servidor para conexões de socket TCP de entrada. Consulte "Soquetes de servidor" na página 832.

Classe	Formatos suportados de dados	Protocolos	Descrição
DatagramSocket (AIR)	Qualquer elemento	UDP	Envia e recebe pacotes UDP. Consulte "Soquetes UDP (AIR)" na página 834

Frequentemente, na criação de aplicativos web é de grande ajuda armazenar informações persistentes sobre o estado do aplicativo do usuário. Páginas e aplicativos HTML geralmente utilizam cookies para este propósito. No Flash Player, você pode utilizar a classe ShareObject para o mesmo propósito. Consulte "Objetos compartilhados" na página 724. (A classe SharedObject pode ser utilizada em aplicativos AIR, mas existem menos restrições ao apenas salvar os dados em um arquivo normal).

Quando o seu Flash Player ou aplicativo do AIR precisa se comunicar com outro Flash Player ou outro aplicativo do AIR no mesmo computador, você poderá utilizar a classe LocalConnection. Por exemplo, dois (ou mais) arquivos SWF na mesma página podem se comunicar. Da mesma forma, um arquivo SWF em uma página pode se comunicar com um aplicativo do AIR. Consulte "Comunicação com outras instâncias do Flash Player e AIR" na página 857.

Quando houver a necessidade de se comunicar com outros processos que não sejam SWF no computador local, você pode utilizar a classe NativeProcess adicionada no AIR 2. A classe NativeProcess permite ao seu aplicativo do AIR lançar e se comunicar com outros aplicativos. Consulte "Comunicação com processos nativos do AIR" na página 864.

Quando for necessário saber informações sobre o ambiente de rede do computador no qual um aplicativo do AIR está sendo executado, pode-se utilizar as seguintes classes:

- NetworkInfo—Fornece informações sobre as interfaces de rede disponíveis, como o endereço IP do computador. Consulte "Interfaces de rede" na página 817.
- DNSResolver—Permite o acesso aos registros de DNS. Consulte "Registro do Sistema de Nome de Domínios (DNS)" na página 821.
- ServiceMonitor—Permite que você monitore a disponibilidade de um servidor. Consulte "Monitoramento de serviços" na página 819.
- URLMonitor—Permite que você monitore a disponibilidade de um recurso em um URL especifico. Consulte "Monitoramento de HTTP" na página 820.
- SocketMonitor e SecureSocketMonitor—Permite que você monitore a disponibilidade de um recurso em um socket. Consulte "Monitoramento de soquetes" na página 820.

Conceitos e termos importantes

A lista a seguir de referência contém termos importantes que você vai encontrar ao programar o código de sistemas de rede e comunicação:

Dados externos Dados que são armazenados em algum formato fora do aplicativo do e carregados no aplicativo quando necessário. Esses dados podem ser armazenados em um arquivo que é carregado diretamente, em um banco de dados ou de algum modo que permita recuperá-los ao chamar scripts ou programas em execução no servidor.

Variáveis codificadas por URL O formato codificado por URL permite representar diversas variáveis (pares de nomes e valores de variáveis) em uma única string de texto. Variáveis individuais são escritas no formato nome=valor. Cada variável (ou seja, cada par de nome e valor) é separada por caracteres E comercial, desta maneira: variável1=valor1&variável2=valor2. Desse modo, é possível enviar um número indefinido de variáveis como uma só mensagem.

Tipo MIME Código padrão usado para identificar o tipo de um dado arquivo em comunicação na Internet. Qualquer tipo de arquivo que tem um código específico usado para identificá-lo. Ao enviar um arquivo ou uma mensagem, um computador (como um servidor Web ou uma ocorrência do Flash Player ou do AIR de usuário) especificará o tipo de arquivo que está sendo enviado.

HTTP forma abreviada do termo em inglês Hypertext Transfer Protocol, um formato padrão para apresentação de páginas da Web e vários outros tipos de conteúdo enviados pela Internet.

Método de solicitação Quando um aplicativo (como um aplicativo do AIR ou um navegador da Web) envia uma mensagem (chamada de solicitação HTTP) para um servidor Web, qualquer dado que está sendo enviado pode ser incorporado à solicitação de duas maneiras, por meio dos métodos de solicitação GET e POST. No servidor, o programa que recebe a solicitação precisará localizar os dados no trecho adequado da solicitação, por isso o método de solicitação usado para enviar dados do seu aplicativo deve corresponder ao método de solicitação usado para ler esses dados no servidor.

Conexão de soquete Uma conexão persistente para comunicação entre dois computadores.

Carregar é o envio de arquivos para outro computador.

Baixar é usado para recuperar um arquivo contido em outro computador.

Interfaces de rede

Adobe AIR 2 e posterior

Você pode usar o objeto NetworkInfo para detectar as interfaces de rede de hardware e software à disposição do seu aplicativo. O objeto NetworkInfo é um objeto *singleton*, e você não precisa criá-lo. Em vez disso, use a propriedade estática de classe `networkInfo` para acessar o único objeto NetworkInfo. Além disso, o objeto NetworkInfo gera um evento `networkChange` quando uma das interfaces disponíveis mudar.

Chame o método `findInterfaces()` para obter uma lista de objetos NetworkInterface. Cada objeto NetworkInterface da lista descreve uma das interfaces disponíveis. O objeto NetworkInterface fornece informações, tais como o endereço IP, endereço de hardware, unidade máxima de transmissão e se a interface está ativa.

O exemplo de código a seguir rastreia as propriedades do NetworkInterface de cada interface no computador cliente:

```
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 );
                }
            }
        }
    }
}
}
```

Para obter mais informações, consulte:

- NetworkInfo
- NetworkInterface
- InterfaceAddress
- Flexpert: Como detectar o tipo de rede com o Flex 4.5

Mudanças na conectividade de rede

Adobe AIR 1.0 e posterior

Seu aplicativo do AIR pode ser executado em ambientes com conectividade de rede incerta e em mudança. Para ajudar um aplicativo a gerenciar conexões em recursos on-line, o Adobe AIR envia um evento de mudança de rede sempre que uma conexão de rede se torna disponível ou indisponível. O objeto NetworkInfo e o objeto NativeApplication do aplicativo geram o evento `networkChange`. Para reagir a esse evento, adicione um ouvinte:

```
NetworkInfo.networkInfo.addEventListener(Event.NETWORK_CHANGE, onNetworkChange);
```

E defina uma função do manipulador de eventos:

```
function onNetworkChange(event:Event)
{
    //Check resource availability
}
```

O evento `networkChange` não indica uma mudança em toda a atividade de rede, mas apenas que uma conexão de rede individual sofreu mudança. O AIR não tenta interpretar o significado da mudança de rede. Um computador em rede pode ter várias conexões reais e virtuais, portanto, perder uma conexão não significa necessariamente perder um recurso. Por outro lado, novas conexões também não garantem melhor disponibilidade de recursos. Às vezes uma nova conexão pode até bloquear o acesso a recursos anteriormente disponíveis (por exemplo, ao se conectar a uma VPN).

Em geral, a única maneira de um aplicativo determinar se ele pode se conectar a um recurso remoto é por tentativa. A estrutura de monitoramento de serviço é um meio, com base em eventos, de reagir a mudanças na conectividade de rede de um host específico.

Nota: *A estrutura de monitoramento de serviço detecta se um servidor responde de maneira aceitável a uma solicitação. Uma verificação bem-sucedida não garante conectividade plena. Serviços dimensionáveis da Web geralmente usam aparatos de cache e balanceamento de carga para redirecionar o tráfego para um cluster de servidores Web. Nessa situação, provedores de serviço só oferecem um diagnóstico parcial de conectividade de rede.*

Monitoramento de serviços

Adobe AIR 1.0 e posterior

O framework do monitor de serviço, separado do framework do AIR, reside no arquivo aircore.swc. Para utilizar o framework, o arquivo aircore.swc deve ser incluído no processo de criação.

O Adobe® Flash® Builder inclui essa biblioteca automaticamente.

A classe ServiceMonitor implementa a estrutura para monitorar serviços de rede e oferece uma funcionalidade básica para monitores de serviço. Por padrão, uma ocorrência da classe ServiceMonitor despacha eventos relacionados à conectividade de rede. O objeto ServiceMonitor envia esses eventos quando a instância é criada e sempre que o tempo de execução detectar uma alteração na rede. Além disso, você pode definir a propriedade `pollInterval` de uma ocorrência ServiceMonitor para verificar a conectividade em um intervalo especificado em milissegundos, independentemente de eventos gerais de conectividade de rede. Um objeto ServiceMonitor não verifica a conectividade de rede até que o método `start()` seja chamado.

A classe URLMonitor, uma subclasse da classe ServiceMonitor, detecta mudanças na conectividade HTTP para um URLRequest especificado.

A classe SocketMonitor, também uma subclasse da classe ServiceMonitor, detecta mudanças na conectividade em um host especificado de uma porta especificada.

Nota: *Em versões anteriores ao AIR 2, o framework do monitor de serviço era publicado na biblioteca servicemonitor.swc. A utilização dessa biblioteca não é mais recomendada. Em vez disso, utilize a biblioteca aircore.swc.*

Flash CS4 e CS5 Professional

Execute as etapas a seguir para utilizar essas classes no Adobe® Flash® CS4 ou CS5 Professional:

1. Selecione o comando Arquivo > Configurações de publicação.
2. Clique no botão Configurações do ActionScript 3.0. Selecione Caminho da biblioteca.
3. Clique no botão Procurar em no SWC e navegue até a pasta AIK no diretório de instalação do Flash Professional.
4. Dentro dessa pasta, localize o arquivo /frameworks/libs/air/aircore.swc (para AIR 2) ou /frameworks/libs/air/servicemonitor.swc (para AIR 1.5).
5. Clique no botão OK.
6. Adicione a seguinte instrução de importação ao código do ActionScript 3.0:

```
import air.net.*;
```

Flash CS3 Professional

Para usar essas classes no Adobe® Flash® CS3 Professional, arraste o componente ServiceMonitorShim do painel Componentes para a Biblioteca e, em seguida, adicione a seguinte declaração `import` ao código do ActionScript 3.0:

```
import air.net.*;
```

Monitoramento de HTTP

Adobe AIR 1.0 e posterior

A classe URLMonitor determina se podem ser feitas solicitações HTTP a um endereço especificado na porta 80 (a porta típica para comunicação HTTP). O código seguinte usa uma instância da classe URLMonitor para detectar mudanças de conectividade no site da Adobe:

```
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);
}
```

Monitoramento de soquetes

Adobe AIR 1.0 e posterior

Aplicativos AIR também podem usar conexões de soquete para conectividade modelo empurrar. Firewalls e roteadores de rede geralmente restringem a comunicação em rede em portas não autorizadas, por questões de segurança. Por esse motivo, os desenvolvedores devem considerar que usuários podem nem sempre ter recursos para fazer conexões de soquete.

O código seguinte usa uma instância da classe SocketMonitor para detectar mudanças de conectividade com uma conexão de soquete. A porta monitorada é a 6667, uma porta comum do IRC:

```
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);
}
```

Caso o servidor socket necessite um conexão segura, você pode utilizar a classe SecureSocketMonitor ao invés da SocketMonitor.

Registro do Sistema de Nome de Domínios (DNS)

Adobe AIR 2.0 e posterior

Você pode consultar os registros do recurso de DNS através da classe DNSResolver. Os registros do recurso de DNS geram informações como o endereço IP de um nome de domínio e o nome de domínio de um endereço IP. Você pode consultar os seguintes tipos de registros do recurso de DNS:

- ARecord—endereço IPv4 de um host.
- AAAARecord—endereço IPv6 de um host.
- MXRecord—registro de troca de email de um host.
- PTRRecord—nome de host de um endereço IP.
- SRVRecord—registro de um serviço.

Para consultar um registro, transmita uma sequência de caracteres de consulta e o objeto de classe que representa o tipo de registro ao método `lookup()` do objeto DNSResolver. A sequência de caracteres de consulta a ser usada depende do tipo de registro:

Classe de registro	Sequência de caracteres de consulta	Exemplo de sequência de caracteres
ARecord	nome do host	“example.com”
AAAARecord	nome do host	“example.com”
MXRecord	nome do host	“example.com”
PTRRecord	Endereço IP	“208.77.188.166”
SRVRecord	Identificador de serviço: _service._protocol.host	“_sip._tcp.example.com”

O seguinte exemplo de código consulta o endereço IP do host “example.com”.

```
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 );
        }
    }
}
```

Para obter mais informações, consulte:

- DNSResolver
- DNSResolverEvent
- ARecord
- AAAARecord
- MXRecord
- PTRRecord
- SRVRecord

Capítulo 43: Soquetes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um soquete é um tipo de conexão de rede estabelecida entre dois processos de computador. Geralmente, os processos são executados em dois computadores diferentes, conectados na mesma rede de Protocolo de Internet (IP). No entanto, os processos conectados podem ser executados no mesmo computador, utilizando o endereço IP especial "local host".

O Adobe Flash Player oferece suporte a soquetes de Protocolo de Controle de Transporte (TCP) do lado do cliente. Um aplicativo Flash Player pode conectar a outro processo, atuando como servidor de soquete, mas não pode aceitar solicitações de conexões de entrada de outros processos. Em outras palavras, um aplicativo Flash Player pode se conectar a um servidor TCP, mas não pode agir como um.

A API do Flash Player também inclui a classe XMLSocket. A classe XMLSocket utiliza um protocolo específico do Flash Player que permite a troca de mensagens XML com um servidor compatível com o protocolo. A classe XMLSocket foi introduzida no ActionScript 1 e ainda possui suporte para fornecer compatibilidade com a versão anterior. No geral, a classe Socket pode ser utilizada para aplicativos novos, a não ser que você esteja estabelecendo uma conexão com um servidor especificamente criado para comunicar-se com Flash XMLSockets.

O Adobe AIR adiciona diversas classes para programação de rede com base em soquetes. Aplicativos AIR podem agir como servidor de soquete TCP com a classe ServerSocket e podem conectar-se a servidores de soquete que requerem segurança SSL ou TLS com a classe SecureSocket. Os aplicativos AIR também podem enviar e receber mensagens de Protocolo de Datagrama Universal (UDP) com a classe DatagramSocket.

Mais tópicos da Ajuda

pacote flash.net

"Conexão a soquetes" na página 1108

Soquetes TCP

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O protocolo de controle de transmissão (TCP) fornece uma maneira de trocar mensagens em uma conexão de rede permanente. O TCP garante que as mensagens enviadas cheguem na ordem correta (salvo quando ocorrem problemas de rede). As conexões TCP requerem um cliente em um servidor. O Flash Player pode criar soquetes de cliente. O Adobe AIR pode, além disso, criar soquetes de servidor.

As APIs de ActionScript a seguir fornecem conexões TCP:

- Soquete — permite que um aplicativo de cliente conecte ao servidor. A classe Socket não pode monitorar conexões de entrada.
- SecureSocket (AIR) —permite que o aplicativo do cliente conecte-se a um servidor confiável e inicie a comunicação criptografada.
- ServerSocket (AIR) — permite que um aplicativo monitore conexões de entrada e atue como servidor.
- XMLSocket — permite que um aplicativo de cliente conecte-se a um servidor XMLSocket.

Soquetes binários de cliente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma conexão de soquete binário é semelhante a um soquete XML, exceto que o cliente e o servidor não ficam limitados à troca de mensagens XML. Em vez disso, a conexão pode transferir dados como informações binárias. Assim, você pode se conectar a uma gama maior de serviços, incluindo servidores de email (POP3, SMTP e IMAP) e novos servidores (NNTP).

Classe Socket

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Socket permite que você faça conexões de soquete e leia e grave dados binários brutos. A classe Socket é útil para interoperar com servidores que usam protocolos binários. Usando conexões de soquete binário, você pode gravar código que interage com vários protocolos de Internet diferentes, como POP3, SMTP, IMAP e NNTP. Essa interação, por sua vez, permite que seus aplicativos se conectem a servidores de email e de notícias.

O Flash Player pode fazer interface com um servidor usando o protocolo binário desse servidor diretamente. Alguns servidores usam a ordem de bytes big-endian, e outros utilizam a ordem de bytes little-endian. A maioria dos servidores na Internet usa a ordem de bytes big-endian porque a "ordem de bytes da rede" é big-endian. A ordem de bytes little-endian é popular porque é a usada pela arquitetura Intel® x86. Você deve usar a ordem de bytes endian compatível com a ordem de bytes do servidor que está enviando ou recebendo dados. Todas as operações executadas pelas interfaces IDataInput e IDataOutput, e as classes que implementam essas interfaces (ByteArray, Socket e URLStream), por padrão, são codificadas no formato big-endian, isto é, com o byte mais significativo primeiro. Esta ordem padrão de bytes foi escolhida para corresponder à ordem de bytes de rede oficial e do Java. Para alterar se a ordem de bytes big-endian ou a little-endian é usada, você pode definir a propriedade `endian` como `Endian.BIG_ENDIAN` ou `Endian.LITTLE_ENDIAN`.

A classe Socket herda todos os métodos definidos pelas interfaces IDataInput e IDataOutput (localizadas no pacote flash.utils). Esses métodos devem ser usados para gravar no Socket e ler a partir do Socket.

Para obter mais informações, consulte:

- Socket
- IDataInput
- IDataOutput
- evento socketData

Soquetes de cliente seguros (AIR)

Adobe AIR 2 e posterior

Você pode usar a classe SecureSocket para conexão a servidores de soquete que usam o Secure Sockets Layer versão 4 (SSLv4) ou o Transport Layer Security versão 1 (TLSv1). Um soquete seguro oferece três benefícios: autenticação do servidor, integridade de dados e sigilo da mensagem. O tempo de execução do autentica um servidor usando o certificado do servidor e seu relacionamento com os certificados raiz ou intermediários da autoridade de certificação no repositório confiável do usuário. O tempo de execução utiliza os algoritmos de criptografia usados pelas implementações de protocolo SSL e TLS para proporcionar integridade de dados e confidencialidade das mensagens.

Quando você se conecta a um servidor através do objeto SecureSocket, o tempo de execução verifica o certificado do servidor utilizando o repositório confiável de certificados. No Windows e Mac, o sistema operacional fornece o armazenamento de confiança. No Linux, o tempo de execução do fornece seu próprio repositório confiável.

Se o certificado do servidor não for válido nem confiável, o tempo de execução gerará um evento `ioError`. Você pode verificar a propriedade `serverCertificateStatus` do objeto SecureSocket para saber por que a verificação falhou. Não há nenhuma previsão para comunicação com um servidor que não possua um certificado válido e confiável.

A classe CertificateStatus define as constantes alfanuméricas que representam os possíveis resultados da verificação:

- Vencido—a data de vencimento do certificado já passou.
- Inválido—há vários motivos pelos quais um certificado pode não ser válido. Por exemplo, o certificado pode ter sido alterado, corrompido, ou pode ser do tipo incorreto.
- Cadeia inválida—um ou mais certificados na cadeia de certificados do servidor não são válidos.
- Discrepância principal—o nome de host do servidor e o nome comum do certificado não coincidem. Em outras palavras, o servidor está usando o certificado incorreto.
- Revogado—a autoridade de certificação emissora revogou o certificado.
- Confiável—o certificado é válido e confiável. Um objeto SecureSocket pode se conectar apenas a um servidor que utilize um certificado válido e confiável.
- Desconhecido—o objeto SecureSocket ainda não verificou o certificado. A propriedade `serverCertificateStatus` tem este valor de status antes que você chame `connect()` e antes que um evento `connect` ou `ioError` seja gerado.
- Signatários não confiáveis—o certificado não gera uma cadeia com um certificado raiz confiável no repositório confiável do computador cliente.

A comunicação com um objeto SecureSocket exige que o servidor use um protocolo seguro e tenha um certificado válido e confiável. Em outros aspectos, usar o objeto SecureSocket é igual a usar um objeto Socket.

O objeto SecureSocket não pode ser usado em todas as plataformas. Use a propriedade `isSupported` da classe SecureSocket para testar se o tempo de execução permite o uso do objeto SecureSocket no computador cliente atual.

Para obter mais informações, consulte:

- SecureSocket
- CertificateStatus
- IDataInput
- IDataOutput
- evento socketData

Exemplo de soquete TCP: construção de um cliente Telnet

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O exemplo de Telnet demonstra técnicas de conexão com um servidor remoto e transmissão de dados usando a classe Socket. O exemplo demonstra as seguintes técnicas:

- Criação de um cliente Telnet personalizado usando a classe Socket
- Envio de texto ao servidor remoto usando um objeto ByteArray
- Tratamento de dados recebidos de um servidor remoto

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Telnet podem ser encontrados na pasta Amostras/Telnet. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
TelnetSocket.fla ou TelnetSocket.mxml	O arquivo principal do aplicativo formado pela interface de usuário para Flex (MXML) ou Flash (FLA).
TelnetSocket.as	Classe Document que fornece a lógica da interface de usuário (somente no Flash).
com/example/programmingas3/Telnet/Telnet.as	Fornece a funcionalidade de cliente Telnet para o aplicativo, como conectar-se a um servidor remoto e enviar, receber e exibir dados.

Visão geral do aplicativo de soquete Telnet

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O arquivo TelnetSocket.mxml principal é responsável por criar a interface do usuário (UI) para o aplicativo inteiro.

Além da UI, esse arquivo também define dois métodos, `login()` e `sendCommand()`, para conectar o usuário ao servidor especificado.

O código a seguir lista o ActionScript no arquivo principal do aplicativo:

```
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 = "";
}
```

A primeira linha de código importa a classe Telnet do pacote com.example.programmingas.socket personalizado. A segunda linha de código declara uma instância da classe Telnet, `telnetClient`, que é inicializada posteriormente pelo método `connect()`. Em seguida, o método `connect()` é declarado e inicializa a variável `telnetClient` declarada anteriormente. Esse método transmite o nome do servidor telnet especificado pelo usuário, a porta do servidor telnet e uma referência a um componente TextArea na lista de exibição que é usada para exibir as respostas de texto do servidor de soquete. As duas linhas finais do método `connect()` definem a propriedade `title` do Painel e ativam o componente Painel, o que permite que o usuário envie dados ao servidor remoto. O método final no arquivo principal do aplicativo, `sendCommand()`, é usado para enviar os comandos do usuário ao servidor remoto como um objeto ByteArray.

Visão geral da classe Telnet

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Telnet é responsável por conectar-se ao servidor Telnet remoto e enviar/receber dados.

A classe Telnet declara as seguintes variáveis particulares:

```
private var serverURL:String;
private var portNumber:int;
private var socket:Socket;
private var ta:TextArea;
private var state:int = 0;
```

A primeira variável, `serverURL`, contém o endereço do servidor especificado pelo usuário ao qual conectar-se.

A segunda variável, `portNumber`, é o número da porta na qual o servidor Telnet está em execução atualmente. Por padrão, o serviço Telnet é executado na porta 23.

A terceira variável, `socket`, é uma instância de Socket que tentará conectar-se ao servidor definido pelas variáveis `serverURL` e `portNumber`.

A quarta variável, `ta`, é uma referência a uma ocorrência do componente TextArea no Palco. Esse componente é usado para exibir respostas do servidor Telnet remoto ou qualquer mensagem de erro possível.

A variável final, `state`, é um valor numérico usado para determinar quais opções são suportadas pelo cliente Telnet.

Conforme visto anteriormente, a função do construtor da classe Telnet é chamada pelo método `connect()` no arquivo principal do aplicativo.

O construtor Telnet utiliza três parâmetros: `server`, `port` e `output`. Os parâmetros `server` e `port` especificam o nome do servidor e o número da porta em que o servidor Telnet está executando. O parâmetro final, `output`, é uma referência a uma instância do componente TextArea no Palco onde a saída do servidor é exibida aos usuários.

```
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();
    }
}
```

Gravação de dados em um soquete

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para gravar dados em uma conexão de soquete, você pode chamar qualquer um dos métodos de gravação da classe Socket. Entre esses métodos de gravação estão os seguintes: `writeBoolean()`, `writeByte()`, `writeBytes()`, `writeDouble()` e outros. Então, esvazie os dados do buffer de saída usando o método `flush()`. No servidor Telnet, os dados são gravados na conexão de soquete usando o método `writeBytes()` que utiliza a matriz de bytes como um parâmetro e envia-a ao buffer de saída. O método `writeBytesToSocket()` é o seguinte:

```
public function writeBytesToSocket(ba:ByteArray):void
{
    socket.writeBytes(ba);
    socket.flush();
}
```

Esse método é chamado pelo método `sendCommand()` do arquivo principal do aplicativo.

Exibição de mensagens do servidor de soquete

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Sempre que uma mensagem é recebida do servidor de soquete, ou ocorre um evento, o método `msg()` personalizado é chamado. Esse método anexa uma sequência de caracteres ao TextArea no Palco e chama um método `setScroll()` personalizado, o que faz com que o componente TextArea role para a parte inferior. O método `msg()` é o seguinte:

```
private function msg(value:String):void
{
    ta.text += value;
    setScroll();
}
```

Se você não rolou automaticamente o conteúdo do componente TextArea, os usuários precisarão arrastar manualmente as barras de rolagem na área de texto para ver a resposta mais recente do servidor.

Rolagem de um componente TextArea

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `setScroll()` contém uma única linha do ActionScript que rola o conteúdo do componente TextArea verticalmente de forma que o usuário possa ver a última linha do texto retornado. O seguinte snippet mostra o método `setScroll()`:

```
public function setScroll():void
{
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}
```

Esse método define a propriedade `verticalScrollPosition`, que é o número da linha superior de caracteres exibida atualmente, e define-a como o valor da propriedade `maxVerticalScrollPosition`.

Soquetes XML

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um soquete XML permite que você crie uma conexão com um servidor remoto que permanece aberta até que ela seja expressamente fechada. Você pode trocar dados de string como, por exemplo, XML entre o servidor e o cliente. Um dos benefícios de utilizar um servidor de soquete XML é que o cliente não precisa solicitar dados expressamente. O servidor pode enviar dados sem aguardar por uma solicitação e enviar dados para cada cliente conectado.

No Flash Player e em conteúdo Adobe AIR fora da área de segurança, as conexões XML requerem a presença de um arquivo de política de soquete no servidor destino. Para obter mais informações, consulte "Controles de site (arquivos de política)" na página 1089 e"Conexão a soquetes" na página 1108.

A classe XMLSocket não pode encapsular em firewalls automaticamente porque, ao contrário do protocolo RTMP (Real-Time Messaging Protocol), ela não tem recurso de encapsulamento HTTP. Se você precisar usar o encapsulamento HTTP, considere o uso do Flash Remoting ou do Flash Media Server (que oferece suporte a RTMP).

As restrições a seguir aplicam-se a como e onde o conteúdo do Flash Player ou de um aplicativo do AIR fora da área de segurança do aplicativo pode utilizar um objeto XMLSocket para conectar-se ao servidor:

- No caso de conteúdo fora da caixa de proteção de segurança do aplicativo, o método `XMLSocket.connect()` só pode se conectar a números de porta TCP maiores ou iguais a 1024. Uma consequência dessa restrição é que os daemons de servidor que se comunicam com o objeto `XMLSocket` também devem ser atribuídos a números de porta maiores ou iguais a 1024. Os números de porta menores que 1024 normalmente são usados por serviços do sistema, como FTP (21), Telnet (23), SMTP (25), HTTP (80) e POP3 (110). Portanto, os objetos XMLSocket são barrados nessas portas por motivos de segurança. A restrição de número de porta limita a possibilidade de esses recursos serem indevidamente acessados e mal utilizados.
- No caso de conteúdo fora da caixa de proteção de segurança do aplicativo, o método `XMLSocket.connect()` só pode se conectar a computadores do mesmo domínio onde reside o conteúdo. (Esta restrição é idêntica às regras de segurança para `URLLoader.load()`.) Para se conectar a um daemon de servidor executado em um domínio que não seja o domínio em que reside o conteúdo, você pode criar um arquivo de políticas entre domínios no servidor que permita acesso de domínios específicos. Para obter detalhes sobre arquivos de políticas entre domínios, consulte "Segurança do AIR" na página 1116.

Nota: *Pode ser desafiador configurar um servidor para comunicar-se com o objeto XMLSocket. Se o seu aplicativo não exigir interatividade em tempo real, use a classe URLLoader em vez da classe XMLSocket.*

Você pode usar os métodos `XMLSocket.connect()` e `XMLSocket.send()` da classe XMLSocket para transferir XML para e de um servidor via conexão de soquete. O método `XMLSocket.connect()` estabelece uma conexão de soquete com uma porta de servidor Web. O método `XMLSocket.send()` analisa um objeto XML para o servidor especificado na conexão de soquete.

Ao chamar o método `XMLSocket.connect()`, o aplicativo abre uma conexão TCP/IP com o servidor de mantém a conexão aberta até que uma das condições a seguir ocorra:

- O método `XMLSocket.close()` da classe XMLSocket é chamado.
- Não há mais referências ao objeto XMLSocket.
- O Flash Player é encerrado.
- A conexão é perdida (por exemplo, o modem se desconecta).

Conexão a um servidor com a classe XMLSocket

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para criar uma conexão de soquete, é necessário criar uma aplicativo server side Flash Player ou AIR para aguardar pela solicitação de conexão do soquete e enviar a resposta para o aplicativo Flash Player ou AIR. Este tipo de aplicativo server side pode ser escrito em AIR ou em outra linguagem de programação como, por exemplo Java, Python ou Perl. Para utilizar a classe XMLSocket, o servidor deve executar uma daemon que entenda o protocolo simples utilizado pela classe XMLSocket:

- Mensagens XML são enviadas por meio de uma conexão de soquete de fluxo TCP/IP full-duplex.
- Cada mensagem XML é um documento XML completo, terminado por um byte zero (0).
- Um número ilimitado de mensagens XML pode ser enviado e recebido por meio de uma única conexão XMLSocket.

Criação e conexão a um servidor de soquete XML Java

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O código a seguir demonstra um servidor XMLSocket simples criado em Java que aceita conexões de entrada e exibe as mensagens recebidas na janela do prompt de comando. Por padrão, um novo servidor é criado na porta 8080 da máquina local, embora você possa especificar um número de porta diferente quando iniciar o servidor a partir da linha de comando.

Crie um novo documento de texto e adicione o seguinte código:

```
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);
    }
}
```

Salve o documento no disco rígido como SimpleServer.java e o compile usando um compilador Java, que cria um arquivo de classe Java chamado SimpleServer.class.

Você pode iniciar o servidor XMLSocket abrindo um prompt de comando e digitando `java SimpleServer`. O arquivo SimpleServer.class pode ser colocado em qualquer lugar do computador local ou da rede; não é necessário colocá-lo no diretório raiz do servidor Web.

Se você não puder iniciar o servidor porque os arquivos não estão localizados dentro do caminho de classe Java, tente iniciar o servidor com `java -classpath. SimpleServer`.

Para se conectar ao XMLSocket do aplicativo , é necessário criar uma nova ocorrência da classe XMLSocket, e chamar o método `XMLSocket.connect()` ao transmitir um nome do host e um número de porta, da seguinte maneira:

```
var xmlsock:XMLSocket = new XMLSocket();
xmlsock.connect("127.0.0.1", 8080);
```

Sempre que você receber dados do servidor, o evento `data` (`flash.events.DataEvent.DATA`) é despachado:

```
xmlsock.addEventListener(DataEvent.DATA, onData);
private function onData(event:DataEvent):void
{
    trace("[" + event.type + "] " + event.data);
}
```

Para enviar dados ao servidor XMLSocket, use o método `XMLSocket.send()` e transmita um objeto ou string XML. O Flash Player converte o parâmetro fornecido em um objeto String e envia o conteúdo ao servidor XMLSocket seguido de um byte zero (0):

```
xmlsock.send(xmlFormattedData);
```

O método `XMLSocket.send()` não retorna um valor que indica se os dados foram transmitidos com êxito. Se houve um erro durante a tentativa de enviar dados, é gerado um erro IOError.

Cada mensagem enviada ao servidor de soquete XML deve ser terminada por uma nova linha (caractere `\n`).

Para obter mais informações, consulte XMLSocket.

Soquetes de servidor

Adobe AIR 2 e posterior

Use a classe ServerSocket para permitir que outros processos se conectem ao seu aplicativo usando um soquete do Protocolo de Controle de Transporte (TCP). O processo em conexão pode estar sendo executado no computador local ou em outro computador conectado à rede. Quando um objeto ServerSocket recebe uma solicitação de conexão, ele gera um evento `connect`. O objeto ServerSocketConnectEvent gerado com o evento contém um objeto Socket. Você pode usar esse objeto Socket em comunicações subsequentes com o outro processo.

Para ouvir conexões de soquete recebidas:

1. Crie um objeto ServerSocket e vincule-o a uma porta local
2. Inclua ouvintes de eventos para o evento `connect`
3. Chame o método `listen()`.
4. Responda ao evento `connect`, que fornece um objeto Socket para cada conexão recebida

O objeto ServerSocket continua a escutar novas conexões até que você chame o método `close()`.

O exemplo de código a seguir ilustra como criar um aplicativo de servidor de soquete. O exemplo ouve conexões que chegam à porta 8087. Quando uma conexão for recebida, o exemplo envia uma mensagem (a sequência de caracteres “Conectado”) para o soquete do cliente. Daí em diante, o servidor "rebate" todas as mensagens para o cliente.

```
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." );
        }
}}
```

Para obter mais informações, consulte:

- ServerSocket
- ServerSocketConnectEvent
- Socket

Soquetes UDP (AIR)

Adobe AIR 2 e posterior

O Protocolo Universal de Datagramas (UDP) é uma forma de troca de mensagens através de uma conexão de rede sem estado (stateless). O UDP não garante que as mensagens serão entregues na ordem nem que elas serão realmente entregues. Com o UDP, o código de rede do sistema operacional normalmente gasta menos tempo organizando, rastreando e confirmando as mensagens. Portanto, as mensagens de UDP normalmente chegam ao aplicativo de destino com um atraso menor do que as mensagens de TCP.

A comunicação por soquete do UDP é útil quando existe a necessidade de enviar informações em tempo real, tais como atualizações de posições em um jogo ou pacotes de som em um aplicativo de bate-papo por voz. Nesses aplicativos, é aceitável alguma perda de dados, e uma baixa latência de transmissão é mais importante que a garantia de chegada. Para praticamente todas as outras finalidades, os soquetes TCP são uma opção mais adequada.

Seu aplicativo do AIR pode enviar e receber mensagens UDP com as classes DatagramSocket e DatagramSocketDataEvent. Para enviar ou receber uma mensagem UDP:

1. Crie um objeto DatagramSocket
2. Adicione um evento ouvinte para o evento `data`
3. Vincule o soquete a um endereço IP e a uma porta local através do método `bind()`
4. Envie mensagens chamando o método `send()`, transmitindo o endereço IP e a porta do computador de destino
5. Receba as mensagens respondendo ao evento `data`. O objeto DatagramSocketDataEvent gerado para este evento contém um objeto ByteArray que contém os dados de mensagem.

O exemplo de código a seguir ilustra como um aplicativo pode enviar e receber mensagens UDP. O exemplo envia uma única mensagem contendo a sequência de caracteres, “Olá.” ao computador de destino. Ele também rastreia o conteúdo de todas as mensagens recebidas.

```
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 ) );
    }
}}
```

Não se esqueça dos seguintes aspectos quando usar soquetes UDP:

- Um único pacote de dados não pode ser maior que a menor unidade máxima de transmissão (MTU) da interface de rede nem que qualquer nó de rede entre o remetente e o destinatário. Todos os dados do objeto ByteArray transmitidos ao método send() são enviados como um único pacote. (No TCP, mensagens grandes são subdivididas em pacotes separados).
- Não há negociação entre o remetente e o destino. As mensagens são descartadas sem erro se o destino não existir ou se não tiver um ouvinte ativo na porta especificada.
- Quando você usar o método `connect()`, as mensagens enviadas de outras fontes serão ignoradas. Uma conexão UDP oferece apenas uma filtragem prática de pacotes. Isso não significa que existe um processo de escuta necessariamente válido no endereço e na porta de destino.
- O tráfego de UDP pode afogar uma rede. Os administradores de rede podem precisar implementar controles de qualidade de serviço caso ocorra um congestionamento de rede. (O TCP tem um controle de tráfego próprio para reduzir o impacto do congestionamento de rede).

Para obter mais informações, consulte:

- DatagramSocket
- DatagramSocketDataEvent
- ByteArray

Endereços IPv6

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Player 9.0.115.0 e suas versões posteriores oferecem suporte a IPv6 (Internet Protocol versão 6). O IPv6 é uma versão do Internet Protocol que oferece suporte a endereços de 128 bits (um aprimoramento do protocolo IPv4 anterior, que oferece suporte a endereços de 32 bits). Talvez seja necessário ativar o IPv6 nas suas interfaces de rede. Para obter mais informações, consulte a Ajuda do sistema operacional que está hospedando os dados.

Se o IPv6 for suportado no sistema de host, será possível especificar endereços literais IPv6 numéricos em URLs, contidos entre parênteses ([]), como apresentado seguir:

```
[2001:db8:ccc3:ffff:0:444d:555e:666f]
```

O Flash Player retorna valores IPv6 literais, de acordo com as regras a seguir:

- O Flash Player retorna o formato longo da string para endereços IPv6.
- O valor de IP não possui abreviações com dois pontos.
- Os dígitos hexadecimais são expressos apenas em letras minúsculas.
- Os endereços IPv6 são apresentados entre parênteses ([]).
- Cada quarteto de endereço é transformado em dígitos hexadecimais de 0 a 4, com os zeros à esquerda omitidos.
- Um quarteto de endereço composto por zeros é transformado em um único zero (e não em dois-pontos duplos), exceto se mencionado na lista de exceções a seguir.

Os valores IPv6 retornados pelo Flash Player têm as seguintes exceções:

- Um endereço IPv6 não especificado (somente zeros) é transformado em [::].
- O endereço IPv6 de retorno ou de host local é transformado em [::1].
- Os endereços IPv4 mapeados (convertido para IPv6) são transformados em [::ffff:a.b.c.d], em que a.b.c.d corresponde a um típico valor decimal IPv4 separado por pontos.
- Os endereços IPv4 compatíveis são transformados em [::a.b.c.d], em que a.b.c.d corresponde a um típico valor decimal IPv4 separado por pontos.

Capítulo 44: Comunicação HTTP

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os aplicativos Adobe® AIR® e Adobe® Flash® Player podem ser comunicar com servidores HTTP para carregar dados, imagens, vídeo e trocar mensagens.

Mais tópicos da Ajuda

flash.net.URLLoader

flash.net.URLStream

flash.net.URLRequest

flash.net.URLRequestDefaults

flash.net.URLRequestHeader

flash.net.URLRequestMethod

flash.net.URLVariables

Carregamento de dados externos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 inclui mecanismos para carregamento de dados de fontes externas. As origens podem fornecer conteúdo estático, como arquivos de texto, ou conteúdo dinâmico, como o conteúdo gerado por um script da Web. Os dados podem ser formatados de diversas maneiras, e o ActionScript oferece a funcionalidade para decodificá-los e acessá-los. Você também pode enviar dados para o servidor externo como parte do processo de recuperação de dados.

Uso da classe URLRequest

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Muitas APIs que carregam dados externos utilizam a classe URLRequest para definir as propriedades da solicitação de rede necessária.

Propriedades de URLRequest

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode definir as propriedades a seguir de uma objeto URLRequest em qualquer área de segurança:

Propriedade	Descrição
contentType	O tipo de conteúdo MIME de qualquer dado enviado com a solicitação de URL. Se nenhum valor é definido para contentType, os valores serão enviados como `application/x-www-form-urlencoded`.
data	Um objeto contendo dados a serem transmitidos com a solicitação de URL.
digest	Uma string que identifica de forma exclusiva o componente da plataforma Adobe assinado a ser armazenado (ou recuperado) no cache do Adobe® Flash® Player.
method	O método de solicitação HTTP, como uma operação GET ou POST. (O conteúdo em execução no domínio de segurança do aplicativo do AIR pode especificar strings diferentes de `"GET"` ou `"POST"` como a propriedade `method`. É permitido qualquer verbo HTTP, e `"GET"` é o método padrão. Consulte "Segurança do AIR" na página 1116.)
requestHeaders	A matriz de cabeçalhos de solicitação HTTP a ser acrescentada à solicitação HTTP. Observe que as permissões para definir alguns cabeçalhos são restritas no Flash Player, assim como no conteúdo do AIR executado fora da área de segurança do aplicativo.
url	Especifica a URL a ser solicitada.

No AIR, você pode definir propriedades adicionais da classe URLRequest, disponíveis somente no conteúdo AIR, executado na área de segurança do aplicativo. O conteúdo na caixa de proteção do aplicativo também pode definir URLs usando novos esquemas de URL (além de esquemas padrão, como `file` e `http`).

Propriedade	Descrição
followRedirects	Especifica se os redirecionamentos devem ser seguidos (`true`, o valor padrão) ou não (`false`). Isso só recebe suporte na caixa de proteção do aplicativo do AIR.
manageCookies	Especifica se a pilha de protocolo HTTP deve gerenciar cookies (`true`, o valor padrão) ou não (`false`) desta solicitação. Definir essa propriedade só recebe suporte na caixa de proteção do aplicativo do AIR.
authenticate	Especifica se as solicitações de autenticação relativas a esta solicitação devem ser manipuladas (`true`). Definir essa propriedade só recebe suporte na caixa de proteção do aplicativo do AIR. O padrão é autenticar as solicitações, o que pode fazer com que seja exibida uma caixa de diálogo de autenticação se o servidor exigir as credenciais. Você também pode definir o nome de usuário e a senha utilizando a classe URLRequestDefaults — consulte "Definir os padrões de URLRequest (somente AIR)" na página 839.
cacheResponse	Especifica se os dados de resposta devem ser armazenados em cache para esta solicitação. Definir essa propriedade só recebe suporte na caixa de proteção do aplicativo do AIR. O padrão é armazenar a resposta em cache (`true`).
useCache	Especifica se o cache local deve ser consultado antes que URLRequest saia em busca dos dados. Definir essa propriedade só recebe suporte na caixa de proteção do aplicativo do AIR. O padrão (`true`) é usar a versão armazenada no cache local, se disponível.
userAgent	Especifica a string user-agent a ser usada na solicitação HTTP.

Nota: *A classe HTMLLoader tem propriedades relacionadas para configurações que pertencem ao conteúdo carregado por um objeto HTMLLoader. Para obter detalhes, consulte "Sobre a classe HTMLLoader" na página 1014 .*

Definir os padrões de URLRequest (somente AIR)

Adobe AIR 1.0 e posterior

A classe URLRequestDefaults permite definir configurações padrão específicas do aplicativo para objetos URLRequest. Por exemplo, o código abaixo define os valores padrão para as propriedades `manageCookies` e `useCache`. Todos os objetos URLRequest novos utilizarão os valores especificados para essas propriedades, em vez dos padrões normais:

```
URLRequestDefaults.manageCookies = false;
URLRequestDefaults.useCache = false;
```

Nota: *A classe URLRequestDefaults é definida para o conteúdo em execução apenas no Adobe AIR. Não há suporte no Flash Player.*

A classe URLRequestDefaults inclui um método `setLoginCredentialsForHost()` que permite especificar um nome de usuário e uma senha padrão para uso em um determinado host. O host, definido no parâmetro hostname do método, pode ser um domínio, como `"www.example.com"`, ou um domínio e um número de porta, como `"www.example.com:80"`. Observe que `"example.com"`, `"www.example.com"` e `"sales.example.com"` são todos considerados hosts únicos.

Essas credenciais só serão utilizadas se exigidas pelo servidor. Se o usuário já foi autenticado (por exemplo, utilizando a caixa de diálogo de autenticação), a chamada do método `setLoginCredentialsForHost()` não altera o usuário autenticado.

O código a seguir define o usuário padrão e a senha utilizada para solicitações enviadas para www.example.com:

```
URLRequestDefaults.setLoginCredentialsForHost("www.example.com", "Ada", "love1816$X");
```

As configurações de URLRequestDefaults somente se aplicam ao domínio atual do aplicativo, com uma exceção. As credenciais enviadas para o método `setLoginCredentialsForHost()` são utilizadas para solicitações feitas em qualquer domínio de aplicativo, dentro do aplicativo do AIR.

Para obter mais informações, consulte a classe URLRequestDefaults em Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Esquemas de URI

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os esquemas padrão de URI como os esquemas a seguir, podem ser utilizados em solicitações feitas a partir de qualquer área de segurança:

http: e https:
Utilize-as para URLs padrão da Internet (da mesma forma como são utilizadas pelo navegador a Web).

arquivo:
Use `file:` para especificar o URL de um arquivo localizado no sistema de arquivos local. Por exemplo:

```
file:///c:/AIR Test/test.txt
```

No AIR, também é possível utilizar os esquemas a seguir para definir o URL para conteúdo em execução na área de segurança do aplicativo:

app:
Use `app:` para especificar um caminho relativo ao diretório raiz do aplicativo instalado. Por exemplo, o seguinte caminho aponta para um subdiretório de recursos do diretório do aplicativo instalado:

```
app:/resources
```

Quando o aplicativo do AIR é executado utilizando o AIR Debug Launcher (ADL), o diretório do aplicativo é o diretório que contém o arquivo descritor do aplicativo.

O URL (e a propriedade `url`) de um objeto File criado com `File.applicationDirectory` utiliza o esquema de URI `app`, como mostrado a seguir:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("assets");
trace(dir.url); // app:/assets
```

app-storage:

Use `app-storage:` para especificar um caminho relativo ao diretório de armazenamento de dados do aplicativo. Para cada aplicativo (e usuário) instalado, o AIR cria um diretório exclusivo de armazenamento do aplicativo, que é um local útil para armazenar dados específicos desse aplicativo. Por exemplo, o caminho a seguir aponta para o arquivo prefs.xml em um subdiretório de configurações do diretório de armazenamento do aplicativo:

```
app-storage:/settings/prefs.xml
```

A URL (e a propriedade `url`) de um objeto File criado com `File.applicationStorageDirectory` usa o esquema de URL `app-storage`, como no exemplo abaixo:

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("prefs.xml");
trace(dir.prefsFile); // app-storage:/prefs.xml
```

mailto:

Você pode usar o esquema mailto em objetos URLRequests enviados para a função `navigateToURL()`. Consulte "Abertura de um URL em outro aplicativo" na página 855.

Você pode usar um objeto URLRequest que utiliza qualquer um desses esquemas de URL para definir uma solicitação de diferentes objetos como, por exemplo, um objeto FileStream ou Sound. Também é possível usar esses esquemas em um conteúdo HTML em execução no AIR; por exemplo, você pode usá-los no atributo `src` de uma tag `img`.

Porém, você só pode usar estes esquemas de URI específicos do AIR, (`app:` e `app-storage:`) no conteúdo localizado na área de segurança do aplicativo. Para obter mais informações, consulte "Segurança do AIR" na página 1116.

Definição de variáveis de URL

Embora seja possível adicionar variáveis diretamente na sequência de caracteres do URL, pode ser mais fácil utilizar a classe URL Variables para definir quaisquer variáveis necessárias para uma solicitação.

Existem três formas em que é possível adicionar parâmetros a um objeto URLVariables:

- No construtor URLVariables
- Com o método `URLVariables.decode()`
- Como propriedades dinâmicas no próprio objeto URLVariables

O exemplo a seguir ilustra todos os três métodos e também como atribuir variáveis ao objeto URLRequest:

```
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;
```

Ao definir variáveis no construtor URLVariables ou no método `URLVariables.decode()`, certifique-se de que os caracteres que possuem um significado especial em uma sequência de caracteres de URI sejam codificados em formato URL. Por exemplo, ao utilizar o & (e comercial) em um nome de parâmetro ou valor, é necessário codificar o &, alterando-o de `&` para `%26`, porque o & atua como um delimitador de parâmetros. A função de nível superior `encodeURIComponent()` pode ser utilizada para isso.

Uso da classe URLLoader

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe URLLoader permite enviar uma solicitação ao servidor e acessar as informações retornadas. Você também pode utilizar a classe URLLoader para acessar arquivos no sistema de arquivos local, onde o acesso de arquivos é permitido (por exemplo, na área de segurança local do Flash Player e do aplicativo do AIR). A classe URLLoader baixa dados de um URL como texto, dados binários ou variáveis codificadas em URL. A classe URLLoader envia eventos como `complete`, `httpStatus`, `ioError`, `open`, `progress` e `securityError`.

O modelo de manipulação de eventos do ActionScript 3.0 é substancialmente diferente do modelo do ActionScript 2.0 que utilizava os manipuladores de evento `LoadVars.onData`, `LoadVars.onHTTPStatus` e `LoadVars.onLoad`. Para obter mais informações sobre como tratar eventos no ActionScript 3.0, consulte "Manipulação de eventos" na página 126

Os dados descarregados só ficam disponíveis depois que o download é concluído. Você pode monitorar o andamento do download (bytes carregados e total de bytes) ouvindo o evento `progress` a ser despachado. No entanto, se um arquivo é carregado muito rapidamente, o evento `progress` pode não ser despachado. Quando um arquivo é baixado com sucesso, o evento `complete` é despachado. Ao definir a propriedade URLLoader `dataFormat`, é possível receber os dados como texto, raw, dados binários ou como objeto URLVariables.

O método `URLLoader.load()` (e opcionalmente o construtor da classe URLLoader) utiliza um único parâmetro, `request`, que é um objeto URLRequest. Um objeto URLRequest contém todas as informações para uma única solicitação HTTP, como o URL de destino, método de solicitação (`GET` ou `POST`), informações de cabeçalho adicionais e o tipo MIME.

Por exemplo, para carregar um pacote XML em um script de servidor, você pode usar o seguinte código:

```
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);
```

O snippet anterior cria um documento XML denominado `dataXML` que contém um pacote XML a ser enviado ao servidor. O exemplo define a propriedade URLRequest `contentType` para `"text/xml"` e atribui o documento XML à propriedade URLRequest `data`. Por fim, o exemplo cria um objeto URLLoader e envia a solicitação para o script remoto utilizando o método `load()`.

Uso da classe URLStream

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe URLStream fornece acesso aos dados descarregados conforme os dados se tornam disponíveis. A classe URLStream também permite encerrar um fluxo antes que o seu download seja finalizado. Os dados descarregados estão disponíveis como dados binários raw.

Ao ler dados de um objeto URLStream, use a propriedade `bytesAvailable` para determinar se dados suficientes estão disponíveis antes da leitura. Uma exceção EOFError é enviada se você tentar ler mais dados do que o disponível.

O evento httpResponseStatus (AIR)

No Adobe AIR, a classe URLStream envia um evento `httpResponseStatus` além do evento `httpStatus`. O evento `httpResponseStatus` é entregue antes de quaisquer dados de resposta. O evento `httpResponseStatus` (representado pela classe HTTPStatusEvent) inclui uma propriedade `responseURL`, que é o URL do qual a resposta foi retornada, e uma propriedade `responseHeaders`, que é uma matriz de objetos URLRequestHeader representando os cabeçalhos de resposta retornados pela resposta.

Carregamento de dados de documentos externos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao construir aplicativos dinâmicos, pode ser útil carregar os dados de arquivos externos ou de scripts do lado do servidor. Isso permite criar aplicativos dinâmicos sem precisar editar ou recompilar seu aplicativo. Por exemplo, se você criar um aplicativo "dica do dia", poderá gravar um script do lado do servidor que recupera uma dica aleatória de um banco de dados e salva-a em um arquivo de texto uma vez por dia. Em seguida, o aplicativo pode carregar o conteúdo de um arquivo de texto estático em vez de consultar o banco de dados a cada vez.

O seguinte snippet cria um objeto URLRequest e URLLoader que carrega o conteúdo de um arquivo de texto externo, params.txt:

```
var request:URLRequest = new URLRequest("params.txt");
var loader:URLLoader = new URLLoader();
loader.load(request);
```

Por padrão, se você não definir um método de solicitação, o Flash Player e o Adobe AIR carregarão o conteúdo usando o método `GET` HTTP. Para enviar os dados usando o método `POST`, defina a propriedade `request.method` como `POST` usando a constante estática `URLRequestMethod.POST`, conforme mostrado no seguinte código:

```
var request:URLRequest = new URLRequest("sendfeedback.cfm");
request.method = URLRequestMethod.POST;
```

O documento externo (params.txt), que é carregado no tempo de execução, contém os seguintes dados:

```
monthNames=January,February,March,April,May,June,July,August,September,October,November,Dece
mber&dayNames=Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

O arquivo contém dois parâmetros: `monthNames` e `dayNames`. Cada parâmetro contém uma lista separada por vírgulas que é analisada como strings. Você pode dividir essa lista em uma matriz usando o método `String.split()`.

Evite usar palavras reservadas ou construções de linguagem como nomes de variáveis em arquivos de dados externos, porque isso dificulta a leitura e a depuração do código.

Depois que os dados são carregados, o evento `complete` é despachado, e o conteúdo do documento externo fica disponível para uso na propriedade `data` de URLLoader, como ilustrado neste código:

```
function completeHandler(event)
{
    var loader2 = event.target;
    air.trace(loader2.data);
}
```

Se o documento remoto contém pares de nome e valor, é possível analisar os dados através da classe URLVariables passando o conteúdo do arquivo carregado, como segue:

```
private function completeHandler(event:Event):void
{
    var loader2:URLLoader = URLLoader(event.target);
    var variables:URLVariables = new URLVariables(loader2.data);
    trace(variables.dayNames);
}
```

Cada par de nome e valor do arquivo externo é criado como propriedade no objeto URLVariables. Cada propriedade no objeto variables do exemplo de código anterior é tratada como string. Se o valor do par de nome e valor é uma lista de termos, você pode converter a string em qualquer matriz chamando o método `String.split()`, como segue:

```
var dayNameArray:Array = variables.dayNames.split(",");
```

Se você estiver carregando dados numéricos de arquivos de texto externos, converta os valores em valores numéricos usando a função de nível superior, como `int()`, `uint()` ou `Number()`.

Em vez de carregar o conteúdo do arquivo remoto como uma string e criar um novo objeto URLVariables, você pode definir a propriedade `URLLoader.dataFormat` como uma das propriedades estáticas encontradas na classe URLLoaderDataFormat. Os três valores possíveis para a propriedade `URLLoader.dataFormat` são os seguintes:

- `URLLoaderDataFormat.BINARY` — a propriedade `URLLoader.data` contém dados binários armazenados em um objeto ByteArray.
- `URLLoaderDataFormat.TEXT` — a propriedade `URLLoader.data` contém texto em um objeto String.
- `URLLoaderDataFormat.VARIABLES` — a propriedade `URLLoader.data` contém variáveis codificadas em URL armazenadas em um objeto URLVariables.

O código a seguir demonstra como a configuração da propriedade `URLLoader.dataFormat` como `URLLoaderDataFormat.VARIABLES` permite analisar automaticamente dados carregados em um objeto URLVariables:

```
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);
        }
    }
}
```

Nota: *O valor padrão de* `URLLoader.dataFormat` *é* `URLLoaderDataFormat.TEXT`*.*

Como ilustrado no exemplo a seguir, carregar XML de um arquivo externo é o mesmo que carregar URLVariables. Você pode criar uma ocorrência de URLRequest e uma de URLLoader e usá-las para baixar um documento XML remoto. Quando o arquivo é baixado completamente, o evento `Event.COMPLETE` é despachado e o conteúdo do arquivo externo é convertido para uma ocorrência XML, que pode ser analisada usando métodos e propriedades XML.

```
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());
        }
    }
}
```

Comunicação com scripts externos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Além de carregar arquivos de dados externos, você pode usar também a classe URLVariables para enviar variáveis a um script do lado do servidor e processar a resposta do servidor. Isso é útil, por exemplo, se você está programando um jogo e deseja enviar a pontuação do usuário para um servidor, que calculará se ela deve ser adicionada à lista de pontuações mais altas, ou mesmo enviar informações de logon de um usuário para um servidor validá-las. Um script de servidor pode processar o nome do usuário e a senha, validá-los em um banco de dados e confirmar se as credenciais fornecidas pelo usuário são válidas.

O seguinte snippet cria um objeto URLVariables denominado `variables` que cria uma nova variável chamada `name`. Em seguida, é criado um objeto URLRequest que especifica a URL do script de servidor para a qual enviar as variáveis. Depois, você define a propriedade `method` do objeto URLRequest para enviar as variáveis como uma solicitação HTTP `POST`. Para adicionar o objeto URLVariables à solicitação de URL, defina a propriedade `data` do objeto URLRequest como o objeto URLVariables criado anteriormente. Por fim, é criada a ocorrência de URLLoader e o método `URLLoader.load()`, que inicia a solicitação, é chamado.

```
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);
}
```

O código a seguir contém o conteúdo do documento greeting.cfm do Adobe ColdFusion® usado no exemplo anterior:

```
<cfif NOT IsDefined("Form.name") OR Len(Trim(Form.Name)) EQ 0>
    <cfset Form.Name = "Stranger" />
</cfif>
<cfoutput>welcomeMessage=#UrlEncodedFormat("Welcome, " & Form.name)#
</cfoutput>
```

Solicitações do serviço da Web

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Existem diversos serviços web com base em HTTP. Os principais tipos são:

- REST
- XML-RPC
- SOAP

Para utilizar um serviço web no ActionScript 3, crie um objeto URLRequest, construa a chamada do serviço web utilizando variáveis URL ou um documento XML e envie o serviço utilizando um objeto URLLoader. O framework do Flex contém diversas classes que facilitam a utilização de serviços web—especialmente úteis para o acesso a serviços SOAP complexos. Desde o Flash Professional CS3, é possível utilizar classes Flex em aplicativos desenvolvidos no Flash Professional e também em aplicativos desenvolvidos no Flash Builder.

Em aplicativos AIR com base em HTML, você pode utilizar as classes URLRequest e URLLoader ou a classe Javascript XMLHttpRequest. Se desejar, você também pode criar uma biblioteca SWF que expõe os componentes de serviço web do framework Flex para o código Javascript.

Quando o aplicativo é executado em um navegador, você somente pode utilizar os serviços web no mesmo domínio de Internet em que o SWF de chamada está localizado, a não ser que o servidor que hospeda o serviço web também hospede um arquivo de política entre domínios que permite o acesso a partir de outro domínio. Uma técnica muito utilizada quando não existe um arquivo de política entre domínios disponível é encaminhar as solicitações por meio de seu próprio servidor. O Adobe Blaze DS E Adobe LiveCycle possuem suporte a encaminhamento de serviço web.

Em aplicativos AIR, um arquivo de políticas entre domínios não é obrigatório quando a chamada do serviço web é originada da área de segurança do aplicativo. O conteúdo do aplicativo do AIR nunca é servido a partir de um domínio remoto, portanto ele não se enquadram nos tipos de ataques que as políticas entre domínios evitam. Em aplicativos AIR com base na web, a área de segurança do aplicativo pode fazer XMLHttpRequests entre domínios. Você também pode permitir o conteúdo em outras áreas de segurança para fazer XMLHttpRequests entre domínios, contanto que o conteúdo seja carregado em um iframe.

Mais tópicos da Ajuda

"Controles de site (arquivos de política)" na página 1089

Adobe BlazeDS

Adobe LiveCycle ES2

Arquitetura REST

XML-RPC

Protocolo SOAP

Solicitações de serviço web estilo REST

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Serviços web estilo REST utilizam verbos do método HTTP para atribuir a ação básica e as variáveis de URL para especificar detalhes de ação. Por exemplo, uma solicitação para obter dados de um item pode utilizar o verbo GET e variáveis de URL para especificar um nome de método e ID de item. A sequência de caracteres do URL resultante pode ser similar a:

```
http://service.example.com/?method=getItem&id=d3452
```

Para acessar um serviço web estilo REST com o ActionScript, você pode utilizar as classes URLRequest, URLVariables e URLLoader classes. No código Javascript de um aplicativo do AIR, você também pode utilizar um XMLHttpRequest.

Geralmente, a programação de uma chamada de serviço web estilo REST no ActionScript envolve as etapas a seguir:

1. Crie um objeto URLRequest.
2. Defina o URL do serviço e o método HTTP no objeto de solicitação.
3. Crie um objeto URLVariables.
4. Configure os parâmetros de chamada do serviço como propriedades dinâmicas do objeto variáveis.
5. Atribua o objeto variáveis à propriedade de dados do objeto de solicitação.
6. Envie uma chamada para o serviço com um objeto URLLoader.
7. Gerencie o evento `complete` enviado pelo URLLoader indicando que a chamada do serviço foi concluída. Também é aconselhável monitorar os diversos eventos de erro que podem ser enviados pelo objeto URLLoader.

Por exemplo, imagine um serviço web que expõe um método de teste que ecoa os parâmetros da chamada de volta para o solicitante. O código ActionScript a seguir pode ser utilizado para chamar um serviço:

```
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 );
}
```

No Javascript de um aplicativo do AIR, você pode fazer a mesma solicitação utilizando o objeto XMLHttpRequest:

```
<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>
```

Solicitações de serviço web XML-RPC

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um serviço web XML-RPC recebe seus parâmetros de chamada como um documento XML, em vez de um conjunto de variáveis do URL. Par efetuar uma operação com um serviço web XML-RPC, crie uma mensagem XML formatada de corretamente e envia-a para o serviço web utilizando o método HTTP `POST`. Além disso, você deve definir o cabeçalho `Content-Type` para a solicitação, de forma que o servidor trate os dados da solicitação como XML.

O exemplo a seguir ilustra como utilizar a mesma chamada de serviço web mostrada no exemplo REST, mas desta vez, como um serviço XML-RPC:

```
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 );
}
```

O WebKit no AIR não oferece suporte a sintaxe E4X de forma que o método utilizado para criar o documento XML no exemplo anterior, não funciona com código Javascript. Em vez disse, utilize os métodos DOM para criar o documento XML ou o documento como uma sequência de caracteres e utilizar a classe Javascript DOMParser para converter a sequência de caracteres para XML.

O exemplo a seguir utiliza métodos DOM para criar uma mensagem XML-RPC e um XMLHttpRequest para executar a operação do serviço web:

```
<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>
```

Solicitação do serviço web SOAP

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O SOAP fortalece o conceito de geral de serviço web XML-RPC e fornece um forma mais rica, no entanto complexa, de transferir dados digitados. Os serviços web SOAP geralmente fornece um arquivo Web Service Description Language (WSDL) que especifica chamadas de serviço web, tipos de dados e o URL do serviço. Embora o ActionScript 3 não forneça suporte direto para SOAP, é possível criar uma mensagem SOAP XML manualmente, publicá-la no servidor e analisar os resultados. No entanto, para outras coisas com exceção do serviço web SOAP mais simples, você pode economizar tempo de desenvolvimento utilizando uma biblioteca SOAP já existente.

O framework Flex inclui biblioteca para acessar serviços web SOAP. No Flash Builder, a biblioteca rpc.swc é incluída automaticamente em projetos Flex, já que ela faz parte do framework Flex. No Flash Professional, você pode adicionar o Flex framework.swc e rpc.swc no caminho da biblioteca de um projeto e, em seguida, acessar as classes Flex com o ActionScript.

Mais tópicos da Ajuda

Utilizando o componente de serviço web Flex no Flash Professional

Cristophe Coenraets: Real-time Trader Desktop para Android

Abertura de um URL em outro aplicativo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar a função `navigateToURL()` para abrir um URL em um navegador da Web ou em outro aplicativo. Para o conteúdo em execução no AIR, a função `navigateToURL()` abre a página no navegador da Web do sistema padrão.

Para o objeto URLRequest que você passar como parâmetro `request` desta função, somente a propriedade `url` é usada.

O primeiro parâmetro da função `navigateToURL()`, o parâmetro `navigate`, é um objeto URLRequest (consulte "Uso da classe URLRequest" na página 838). O segundo é um parâmetro `window` opcional, no qual você pode especificar o nome da janela. Por exemplo, o código a seguir abre a página www.adobe.com:

```
var url:String = "http://www.adobe.com";
var urlReq:URLRequest = new URLRequest(url);
navigateToURL(urlReq);
```

Nota: *Na utilização da função `navigateToURL()`, o tempo de execução trata um objeto URLRequest que usa o método POST (um que tenha a propriedade `method` definida como `URLRequestMethod.POST`)*

Quando é usada a função `navigateToURL()`, são permitidos esquemas de URL baseados na área de segurança do código que chama a função `navigateToURL()`.

Algumas APIs permitem iniciar conteúdo em um navegador da Web. Por motivos de segurança, alguns esquemas de URL são proibidos quando essas APIs são utilizadas no AIR. A lista de esquemas proibidos depende da caixa de proteção de segurança do código que usa a API. (Para obter detalhes sobre caixas de proteção de segurança, consulte "Segurança do AIR" na página 1116.)

Caixa de proteção do aplicativo (somente AIR)

Qualquer esquema URI pode ser utilizado no URL lançado pelo conteúdo em execução em uma caixa de segurança de um aplicativo do AIR. Um aplicativo deve ser registrado para manipular o esquema URI; caso contrário, a solicitação não será atendida. Os seguintes esquemas são suportados em muitos computadores e dispositivos:

- `http:`
- `https:`
- `arquivo:`
- `mailto:` – O AIR direciona essas solicitações para o aplicativo de email registrado sistema
- `sms:` — O AIR conduz as solicitações `sms:` para o aplicativo de mensagem de texto padrão. O formato do URL precisa estar em conformidade com as convenções do sistema em que o aplicativo está sendo executado. Por exemplo, no Android, o esquema URI precisa estar em minúsculas.

  ```
  navigateToURL( new URLRequest( "sms:+15555550101") );
  ```

- `tel:` — O AIR conduz as solicitações `tel:` para o aplicativo de discagem de telefone padrão. O formato do URL precisa estar em conformidade com as convenções do sistema em que o aplicativo está sendo executado. Por exemplo, no Android, o esquema URI precisa estar em minúsculas.

  ```
  navigateToURL( new URLRequest( "tel:5555555555") );
  ```

- `market:` — O AIR conduz as solicitações`market:` ao aplicativo Market geralmente suportado nos dispositivos do Android.

```
navigateToURL( new URLRequest( "market://search?q=Adobe Flash") );
navigateToURL( new URLRequest( "market://search?q=pname:com.adobe.flashplayer") );
```

Onde for permitido pelo sistema operacional, os aplicativos poderão definir e registrar esquemas URI customizados. É possível criar um URL utilizando o esquema para lançar o aplicativo a partir do AIR.

Caixas de proteção remotas

Os esquemas abaixo são permitidos. Use esses esquemas da mesma forma que você os utilizaria em um navegador da Web.

- `http:`
- `https:`
- `mailto:` – O AIR direciona essas solicitações para o aplicativo de email registrado sistema

Todos os demais esquemas de URI são proibidos.

Caixa de proteção Local com arquivo

Os esquemas abaixo são permitidos. Use esses esquemas da mesma forma que você os utilizaria em um navegador da Web.

- `arquivo:`
- `mailto:` – O AIR direciona essas solicitações para o aplicativo de email registrado sistema

Todos os demais esquemas de URI são proibidos.

Caixa de proteção Local com rede

Os esquemas abaixo são permitidos. Use esses esquemas da mesma forma que você os utilizaria em um navegador da Web.

- `http:`
- `https:`
- `mailto:` – O AIR direciona essas solicitações para o aplicativo de email registrado sistema

Todos os demais esquemas de URI são proibidos.

Caixa de proteção Local confiável

Os esquemas abaixo são permitidos. Use esses esquemas da mesma forma que você os utilizaria em um navegador da Web.

- `arquivo:`
- `http:`
- https:
- `mailto:` – O AIR direciona essas solicitações para o aplicativo de email registrado sistema

Todos os demais esquemas de URI são proibidos.

Capítulo 45: Comunicação com outras instâncias do Flash Player e AIR

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe LocalConnection permite a comunicação entre aplicativos Adobe® AIR® e também com o conteúdo SWF executado no navegador. Você também pode usar a classe LocalConnection para se comunicar entre um aplicativo do AIR e o conteúdo SWF em execução no navegador. A classe LocalConnection permite construir aplicativos versáteis que podem compartilhar dados entre as instâncias do Flash Player e do AIR.

Sobre a classe Local

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe LocalConnection permite desenvolver arquivos SWF que podem enviar instruções a outros arquivos SWF sem o uso do método fscommand() ou do JavaScript. Objetos LocalConnection podem se comunicar apenas entre arquivos SWF que estão em execução no mesmo computador cliente, mas podem ser executados em diferentes aplicativos. Por exemplo, um arquivo SWF em execução em um navegador e um arquivo SWF em execução em um projetor podem compartilhar informações, com o projetor mantendo informações locais e o arquivo SWF com base em navegador conectado remotamente. (Um projetor é um arquivo SWF salvo em um formato que pode ser executado como um aplicativo autônomo, isto é, o projetor não requer que o Flash Player seja instalado porque está incorporado dentro do executável.)

Objetos LocalConnection podem ser usados para comunicação entre SWFs usando diferentes versões do ActionScript:

- Objetos LocalConnection do ActionScript 3.0 podem se comunicar com objetos LocalConnection criados no ActionScript 1.0 ou 2.0.
- Objetos LocalConnection do ActionScript 1.0 ou 2.0 podem se comunicar com objetos LocalConnection criados no ActionScript 3.0.

O Flash Player manipula automaticamente essa comunicação entre objetos LocalConnection de diferentes versões.

O modo mais simples de usar um objeto LocalConnection é permitir a comunicação somente entre objetos LocalConnection localizados no mesmo domínio ou no mesmo aplicativo do AIR. Desse modo, não é necessário se preocupar com problemas de segurança. No entanto, se precisar permitir a comunicação entre domínios, há vários modos de implementar medidas de segurança. Para obter mais informações, consulte a discussão do parâmetro `connectionName` do método `send()` e das entradas `allowDomain()` e `domain` na listagem da classe LocalConnection, na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

É possível usar objetos LocalConnection para enviar e receber dados dentro de um arquivo SWF único, mas a Adobe não recomenda esse procedimento. Em vez disso, use objetos compartilhados.

Há três maneiras de adicionar métodos de retorno de chamada aos objetos LocalConnection:

- Criar subclasse da classe LocalConnection e adicionar métodos.
- Definir a propriedade `LocalConnection.client` como um objeto que implementa os métodos.
- Criar uma classe dinâmica que estende o LocalConnection e anexar métodos dinamicamente.

A primeira maneira de adicionar métodos de retorno de chamada é estender a classe LocalConnection. Você define os métodos dentro da classe personalizada em vez de adicioná-los dinamicamente à ocorrência de LocalConnection. Essa abordagem é demonstrada no código a seguir:

```
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);
        }
    }
}
```

É possível utilizar o código a seguir para criar uma nova instância da classe CustomLocalConnection:

```
var serverLC:CustomLocalConnection;
serverLC = new CustomLocalConnection("serverName");
```

A segunda maneira de adicionar métodos de retorno de chamada é usar a propriedade `LocalConnection.client`. Isso envolve a criação de uma classe personalizada e a atribuição de uma nova ocorrência para a propriedade `cliente`, conforme mostrado no código a seguir:

```
var lc:LocalConnection = new LocalConnection();
lc.client = new CustomClient();
```

A propriedade `LocalConnection.client` indica os métodos de retorno de chamada do objeto que devem ser chamados. No código anterior, a propriedade `client` foi definida como uma nova ocorrência de uma classe personalizada, CustomClient. O valor padrão da propriedade `client` é a ocorrência LocalConnection atual. Você poderá usar a propriedade `client`, se tiver dois manipuladores de dados que tenham o mesmo conjunto de métodos, mas funcionam de modo diferente, por exemplo, em um aplicativo em que um botão em uma janela alterna a exibição em uma segunda janela.

Para criar a classe CustomClient, você pode usar o seguinte código:

```
package
{
    public class CustomClient extends Object
    {
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

A terceira maneira de adicionar métodos de retorno de chamada criando uma classe dinâmica e anexando os métodos dinamicamente, é muito semelhante ao uso da classe LocalConnection em versões anteriores do ActionScript, conforme mostrado no código a seguir:

```
import flash.net.LocalConnection;
dynamic class DynamicLocalConnection extends LocalConnection {}
```

Os métodos de retorno de chamada podem ser adicionados dinamicamente a essa classe usando o seguinte código:

```
var connection:DynamicLocalConnection = new DynamicLocalConnection();
connection.onMethod = this.onMethod;
// Add your code here.
public function onMethod(timeString:String):void
{
    trace("onMethod called at: " + timeString);
}
```

A maneira anterior de adicionar métodos de retorno de chamada não é recomendada porque o código não é muito portátil. Além disso, o uso desse método de criação de conexões locais pode criar problemas de desempenho, porque o acesso a propriedades dinâmicas é drasticamente mais lento do que o acesso a propriedades seladas.

Propriedade isPerUser

A propriedade `isPerUser` foi adicionada ao Flash Player (10.0.32) e AIR (1.5.2) para solucionar um conflito que ocorre quando mais de um usuário está conectado em um computador Mac. Em outros sistemas, a propriedade é ignorada, já que a conexão local sempre foi estendida aos usuários. A propriedade `isPerUser` deve ser definida para `true` no código novo. No entanto, o valor padrão é `false` para manter a compatibilidade com versões anteriores. O padrão pode ser mudado em versões futuras dos tempos de execução.

Envio de mensagens entre dois aplicativos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use a classe LocalConnection para se comunicar entre aplicativos AIR diferentes e entre aplicativos AIR diferentes e aplicativos Adobe® Flash® Player (SWF) diferentes em execução em um navegador. Você também pode usar a classe LocalConnection para se comunicar entre um aplicativo do AIR e um aplicativo SWF em execução no navegador.

Por exemplo, você pode ter várias ocorrências do Flash Player em uma página da Web ou fazer com que uma ocorrência do Flash Player recupere dados de uma ocorrência do Flash Player em uma janela pop-up.

O código seguinte define um objeto LocalConnection que atua como um servidor e aceita chamadas LocalConnection recebidas de outros aplicativos:

```
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");
            }
        }
    }
}
```

Esse código primeiro cria um objeto LocalConnection chamado `lc` e define a propriedade `client` como um objeto, `clientObject`. Quando outro aplicativo chama um método nessa instância LocalConnection, o tempo de execução pesquisa por esse método no objeto `clientObject`.

Caso possua uma conexão com o nome especificado, uma exceção de Argument Error é enviada, indicando que a tentativa de conexão falhou porque o objeto já está conectado.

Sempre que uma ocorrência do Flash Player se conecta a esse arquivo SWF e tenta chamar qualquer método para a conexão local especificada, a solicitação é enviada para a classe especificada pela propriedade `client`, definida como classe CustomClient1:

```
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");
        }
    }
}
```

Para criar um servidor LocalConnection, chame o método `LocalConnection.connect()` e forneça um nome de conexão exclusivo. Se você já tiver uma conexão com o nome especificado, um erro ArgumentError será gerado, indicando que houve falha durante a tentativa de conexão, porque o objeto já está conectado.

O snippet seguinte demonstra como criar uma LocalConnection com o nome `conn1`:

```
try
{
    connection.connect("conn1");
}
catch (error:ArgumentError)
{
    trace("Error! Server already exists\n");
}
```

A conexão a um aplicativo primário a partir de um secundário requer primeiramente que você crie um objeto LocalConnection no objeto LocalConnection de envio e, em seguida, chame o método `LocalConnection.send()` com o nome da conexão e o nome do método a ser executado. Por exemplo, para enviar o método `doQuit` para o objeto LocalConnection que você criou anteriormente, use o seguinte código:

```
sendingConnection.send("conn1", "doQuit");
```

Esse código se conecta a um objeto LocalConnection existente com o nome de conexão `conn1` e chama o método `doMessage()` no aplicativo remoto. Para enviar parâmetros para o aplicativo remoto, especifique argumentos adicionais após o nome do método, no método `send()`, como mostrado no seguinte snippet:

```
sendingConnection.send("conn1", "doMessage", "Hello world");
```

Conexão a conteúdo em domínios diferentes e a aplicativos AIR

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para permitir conexões apenas de domínios específicos, chame o método `allowDomain()` ou o `allowInsecureDomain()` da classe LocalConnection e passe uma lista de um ou mais domínios que são permitidos para acessar esse objeto LocalConnection, passando um ou mais nomes de domínios a serem permitidos.

Em versões anteriores do ActionScript, `LocalConnection.allowDomain()` e `LocalConnection.allowInsecureDomain()` eram métodos de retorno de chamada que precisavam ser implementados por desenvolvedores e retornar um valor booliano. No ActionScript 3.0, `LocalConnection.allowDomain()` e `LocalConnection.allowInsecureDomain()` são métodos internos que os desenvolvedores podem chamar exatamente como `Security.allowDomain()` e `Security.allowInsecureDomain()`, transmitindo um ou mais nomes de domínios a serem permitidos.

O Flash Player 8 introduziu restrições de segurança em arquivos SWF locais. Um arquivo SWF com acesso permitido à Internet não pode ter acesso também ao sistema de arquivos local. Se você especificar `localhost`, qualquer arquivo SWF local poderá acessar o arquivo SWF. Se o método `LocalConnection.send()` tentar se comunicar com um arquivo SWF a partir de uma caixa de proteção de segurança à qual o código de chamada não possui acesso, um evento `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) será despachado. Para solucionar esse erro, você pode especificar o domínio do chamador no método `LocalConnection.allowDomain()` do receptor.

Há dois valores especiais que você pode passar para os métodos `LocalConnection.allowDomain()` e `LocalConnection.allowInsecureDomain()`: `*` e `localhost`. O valor asterisco `(*)` permite o acesso de todos os domínios. A string `localhost` permite chamadas para o aplicativo a partir do conteúdo instalado localmente, mas fora do diretório de recursos do aplicativo.

Se o método `LocalConnection.send()` tentar se comunicar com um aplicativo a partir de uma caixa de proteção de segurança à qual o código de chamada não tem acesso, um evento `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) será despachado. Para solucionar esse erro, você pode especificar o domínio do chamador no método `LocalConnection.allowDomain()` do receptor.

Se você implementar a comunicação apenas entre o conteúdo no mesmo domínio, é possível especificar um parâmetro `connectionName` que não inicie com um caractere sublinhado (_) e não especifique um nome de domínio (por exemplo, `myDomain:connectionName`). Use a mesma string no comando `LocalConnection.connect(connectionName)`.

Se você implementar a comunicação entre o conteúdo em domínios diferentes, especifique um parâmetro `connectionName` que inicie com um sublinhado. Especificar o sublinhado torna o conteúdo com o objeto LocalConnection de recebimento mais durável entre domínios. Estes são os dois casos possíveis:

- Se a string para `connectionName` não começar com um sublinhado, o tempo de execução adicionará um prefixo com o superdomínio e um caractere de dois pontos (por exemplo, `myDomain:connectionName`). Embora isso garanta que a sua conexão não entre em conflito com as conexões de nome idêntico em outros domínios, qualquer objeto LocalConnection de envio deve especificar esse superdomínio (por exemplo, `myDomain:connectionName`). Se você mover o arquivo HTML ou SWF com o objeto LocalConnection de recebimento para outro domínio, o tempo de execução alterará o prefixo de para refletir o novo superdomínio (por exemplo, `anotherDomain:connectionName`). Todos os objetos LocalConnection de envio precisam ser manualmente editados de modo a apontar para o novo superdomínio.
- Se a string para `connectionName` começar com um sublinhado (por exemplo, `_connectionName`), o tempo de execução não adicionará um prefixo a essa string. Isso significa que os objetos LocalConnection de recebimento e envio usam strings idênticas para `connectionName`. Se o objeto de recebimento usar `LocalConnection.allowDomain()` para especificar que serão aceitas conexões de qualquer domínio, você poderá mover o arquivo HTML OU SWF com o objeto LocalConnection de recebimento para outro domínio sem alterar nenhum objeto LocalConnection de envio.

  Uma desvantagem de usar nomes sublinhados em `connectionName` é o potencial para colisões, como quando dois aplicativos tentam, ambos, se conectar, usando o mesmo `connectionName`. Uma outra desvantagem está relacionada à segurança. Nomes de conexões que utilizam sintaxe sublinhada não identificam o domínio do aplicativo ouvinte. Por esses motivos, nomes qualificados de domínio são preferíveis.

Adobe AIR

Para comunicar-se com conteúdo executado na área de segurança de aplicativo do AIR (conteúdo instalado por aplicativo do AIR), você deve utilizar um prefixo no nome da conexão com um superdomínio que identifica o aplicativo do AIR. A string de superdomínio inicia com `app#` seguido pelo ID de aplicativo ID e um ponto (.) , seguido pelo ID do publicador (se definido). Por exemplo, o superdomínio apropriado a ser utilizado com o parâmetro `connectionName` para um aplicativo com o ID de aplicativo, `com.example.air.MyApp` e ID de publicador é: `"app#com.example.air.MyApp"`. Portanto, se o nome de conexão base for “appConnection,” a string inteira a ser utilizada no parâmetro `connectionName` é: `"app#com.example.air.MyApp:appConnection"`. Se o aplicativo possuir o ID de publicador, o ID também deve ser incluído na string do superdomínio: `"app#com.example.air.MyApp.B146A943FBD637B68C334022D304CEA226D129B4.1"`.

Quando você permite que outro aplicativo do AIR se comunique com seu aplicativo pela conexão local, é preciso chamar o `allowDomain()` do objeto LocalConnection, passando o nome de domínio da conexão local. Para um aplicativo do AIR, esse nome de domínio é formado a partir das IDs do aplicativo e do editor, da mesma forma que na string de conexão. Por exemplo, se o aplicativo do AIR de envio tem uma ID de aplicativo de `com.example.air.FriendlyApp` e uma ID de editor de `214649436BD677B62C33D02233043EA236D13934.1`, a string de domínio a usar para permitir que esse aplicativo se conecte é: `app#com.example.air.FriendlyApp.214649436BD677B62C33D02233043EA236D13934.1`. (Desde a versão 1.5.3 do AIR, nem todos os aplicativos AIR têm ID de publicador.)

Capítulo 46: Comunicação com processos nativos do AIR

Adobe AIR 2 e posterior

Como o Adobe AIR 2, os aplicativos AIR podem executar e se comunicar com outros processos nativos por meio da linha de comando. Por exemplo, um aplicativo do AIR pode executar um processo e se comunicar com ele pode meio da entrada padrão e dos fluxos de saída.

Para se comunicar com processos nativos, empacote um aplicativo do AIR para ser instalado através de um instalador nativo. O tipo de arquivo do instalador nativo é específico do sistema operacional para o qual é criado:

- É um arquivo DMG no Mac OS.
- É um arquivo EXE no Windows.
- É um pacote RPM ou DEB no Linux.

Esses aplicativos são conhecidos como aplicativos de perfil estendido de desktop. Você pode criar um arquivo do instalador nativo especificando a opção `-target native` ao chamar o comando `-package` usando ADT.

Mais tópicos da Ajuda

flash.filesystem.File.openWithDefaultApplication()

flash.desktop. NativeProcess

Visão geral das comunicações do processo nativo

Adobe AIR 2 e posterior

Uma aplicativo do AIR no perfil estendido de desktop pode executar um arquivo, como se ele fosse chamado pela linha de comando. Ele pode se comunicar com os streams padrão do processo nativo. Os fluxos padrão incluem o fluxo de entrada padrão (stdin), o fluxo de saída (stdout) e o fluxo de erro padrão (stderr).

Nota: *Os aplicativos no perfil estendido de desktop também podem abrir arquivos e aplicativos usando o método* `File.openWithDefaultApplication()`*. No entanto, usar este método não fornece ao aplicativo do AIR acesso aos fluxos padrão. Para obter mais informações, consulte "Abertura de arquivos com o aplicativo padrão do sistema" na página 703.*

O exemplo de código a seguir mostra como abrir um aplicativo test.exe no diretório do aplicativo. O aplicativo passa o argumento `"hello"` como argumento da linha de comando, e acrescenta um ouvinte de evento ao fluxo de saída padrão do processo:

```
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);
}
```

Abertura e fechamento de um processo nativo

Adobe AIR 2 e posterior

Para abrir um processo nativo, configure um objeto NativeProcessInfo para fazer o seguinte:

- Aponte para o arquivo que você deseja abrir
- Armazene os argumentos da linha de comando a serem passados para o processo quando abertos (opcional)
- Defina o diretório de trabalho do processo (opcional)

Para iniciar o processo nativo, passe o objeto NativeProcessInfo como parâmetro do método `start()` de um objeto NativeProcess.

Por exemplo, o código a seguir mostra como abrir um aplicativo test.exe no diretório do aplicativo. O aplicativo passa o argumento `"hello"` e define o diretório de documentos do usuário como diretório de trabalho:

```
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);
```

Para encerrar o processo, chame o método `exit()` do objeto NativeProcess.

Se você desejar que um arquivo seja executável no aplicativo instalado, certifique-se de que ele é executável no sistema de arquivos, durante a criação do pacote do aplicativo. (No Mac e Linux, você pode utilizar o comando chmod para definir o indicador de executável, se necessário.)

Comunicação com um processo nativo

Adobe AIR 2 e posterior

Depois que um aplicativo do AIR iniciar um processo nativo, ele pode se comunicar com os fluxos de entrada padrão, saída padrão e erro padrão do processo.

Você lê e grava os dados nos fluxos usando as seguintes propriedades do objeto NativeProcess:

- `standardInput` — Contém acesso aos dados do fluxo de entrada padrão.
- `standardOutput` — Contém acesso aos dados do fluxo de saída padrão.
- `standardError` — Contém acesso aos dados do fluxo de erro padrão.

Gravação no fluxo de entrada padrão

Você pode gravar os dados no fluxo de entrada padrão usando os métodos de gravação da propriedade `standardInput` do objeto NativeProcess. Como o aplicativo do AIR grava os dados ao processo, o objeto NativeProcess despacha os eventos `standardInputProgress`.

Se ocorrer um erro na gravação para um fluxo de entrada padrão, o objeto NativeProcess envia um evento `ioErrorStandardInput`.

Você pode fechar o fluxo de entrada chamando o método `closeInput()` do objeto NativeProcess. Quando o fluxo de entrada se fecha, o objeto NativeProcess despacha o evento `standardInputClose`.

```
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();
}
```

Leitura de um fluxo de saída padrão

Você pode ler os dados do fluxo de saída padrão usando os métodos de leitura desta propriedade. Como o aplicativo do AIR recebe os dados do fluxo de saída do processo, o objeto NativeProcess despacha os eventos `standardOutputData`.

Se ocorrer um erro ao gravar no fluxo de saída padrão, o objeto NativeProcess envia um evento `standardOutputError`.

Quando o processo fecha o fluxo de saída, o objeto NativeProcess envia um evento `standardOutputClose`.

Ao ler dados do stream de entrada padrão, certifique-se de ler os dados como eles são gerados. Em outras palavras, adicione um ouvinte de eventos para o evento `standardOutputData`. No ouvinte de eventos `standardOutputData` , leia os dados da propriedade `standardOutput` do objeto NativeProcess. Não fique apenas aguardando pelo evento `standardOutputClose` ou pelo evento `exit` para ler todos os dados. Caso você não leia os dados na medida em que o processo nativo os gera, o buffer pode encher ou dados podem ser perdidos. Um buffer cheio pode fazer com que o processo nativo pare quando estiver tentando escrever mais dados. No entanto, se você não registrar um ouvinte de eventos para o evento `standardOutputData` , o buffer não vai encher e o processo não vai parar. Neste caso, os dados não poderão ser acessados.

```
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);
}
```

Leitura de um fluxo de erro padrão

Você pode ler os dados do fluxo de erro padrão usando os métodos de leitura desta propriedade. Como o aplicativo do AIR lê os dados de erros do fluxo de erro do processo, o objeto NativeProcess despacha os eventos `standardErrorData`.

Se ocorrer um erro ao gravar no fluxo de erro padrão, o objeto NativeProcess envia um evento `standardErrorIoError`.

Quando o processo fecha o fluxo de erros, o objeto NativeProcess envia um evento `standardErrorClose`..

Ao ler dados do stream de erro padrão, certifique-se de ler os dados como eles são gerados. Em outras palavras, adicione um ouvinte de eventos para o evento `standardErrorData`. No ouvinte de eventos `standardErrorData` , leia os dados da propriedade `standardError` do objeto NativeProcess. Não fique apenas aguardando pelo evento `standardErrorClose` ou pelo evento `exit` para ler todos os dados. Caso você não leia os dados na medida em que o processo nativo os gera, o buffer pode encher ou dados podem ser perdidos. Um buffer cheio pode fazer com que o processo nativo pare quando estiver tentando escrever mais dados. No entanto, se você não registrar um ouvinte de eventos para o evento `standardErrorData` , o buffer não vai encher e o processo não vai parar. Neste caso, os dados não poderão ser acessados.

```
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);
}
```

Considerações de segurança para a comunicação do processo nativo

Adobe AIR 2 e posterior

A API de processo nativa pode executar qualquer executável no sistema do usuário. Seja muito cuidadoso ao construir e executar comandos. Se qualquer parte de um comando a ser executado tiver origem em uma fonte externa, valide cuidadosamente o comando para execução. Da mesma forma, o aplicativo do AIR deve validar quaisquer dados enviados para um processo em execução.

No entanto, a validação de entrada pode ser difícil. Para evitar problemas, o ideal é escrever um aplicativo nativo (ex: arquivo EXE do Windows) que possui a API específica. Estas APIs devem processar só aqueles comandos definidos pelo aplicativo. Por exemplo, o aplicativo pode aceitar somente um conjunto limitado de instruções por meio de fluxo de entrada padrão.

O AIR no Windows não permite que você execute arquivos do tipo .bat diretamente. O aplicativo interpretador de comandos (cmd.exe) executa os arquivos .bat do Windows. Ao chamar um arquivo .bat, este aplicativo de comando pode interpretar argumentos enviados ao comando como, por exemplo, aplicativos adicionais a serem executados. Uma injeção maliciosa de caracteres adicionais na string de argumento pode fazer com que o cmd.exe execute um aplicativo não seguro e que pode causar danos. Por exemplo, sem a validação apropriada de dados, seu aplicativo do AIR pode fazer uma chamada para `myBat.bat myArguments c:/evil.exe`. O aplicativo de comando iniciaria o aplicativo evil.exe além de executar o arquivo de lote.

Capítulo 47: Uso da API externa

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A API externa do ActionScript 3.0 (flash.external.ExternalInterface) permite a comunicação direta entre o ActionScript e o aplicativo de contêiner no qual o Adobe Flash Player está sendo executado. Use a API ExternalInterface para criar uma interação entre um documento SWF e o JavaScript em uma página HTML.

Você pode utilizar a API externa para interagir com um aplicativo de contêiner, como transmitir dados entre o ActionScript e o JavaScript em uma página HTML.

Algumas tarefas comuns da API externa são:

- Obtenção de informações sobre o aplicativo de contêiner
- Utilização do ActionScript para chamar o código em uma página da Web exibida em um navegador ou em um aplicativo de desktop do AIR
- Chamada do código do ActionScript de uma página da Web
- Criação de um proxy para simplificar a chamada do código do ActionScript de uma página da Web

Nota: *Esta discussão abrange apenas a comunicação entre o ActionScript em um arquivo SWF e o aplicativo de contêiner que inclui uma referência ao Flash Player ou à ocorrência na qual o arquivo SWF está carregado. Qualquer outro uso do Flash Player em um aplicativo está fora do escopo desta documentação. O Flash Player foi desenvolvido para ser usado como um plug-in de navegador ou como um projetor (aplicativo dedicado). Outros ambientes de uso podem ter suporte limitado.*

Utilizando a API externa no AIR

Como um aplicativo do AIR não possui um contêiner externo, esta interface externa geralmente não se aplica - ou não é geralmente necessária. Quando seu aplicativo do AIR carrega um arquivo SWF diretamente, o código do aplicativo pode se comunicar diretamente como o código do ActionScript no arquivo SWF (sujeito a restrições de caixa de proteção de segurança).

No entanto, quando seu aplicativo do AIR carrega um arquivo SWF utilizando uma página HTML em um objeto HTMLLoader (ou um componente HTML no Flex) o objeto HTMLLoader serve como o contêiner externo. Portanto, você pode utilizar a interface externa para estabelecer comunicação entre o código ActionScript no SWF carregado e o código JavaScript na página HTML carregado no HTMLLoader.

Noções básicas de uso da API externa

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Embora em alguns casos um arquivo SWF possa funcionar sozinho (por exemplo, se você usar o Adobe® Flash® Professional para criar um projetor SWF), na maioria dos casos, o aplicativo SWF funciona como elemento dentro de outro aplicativo. Normalmente, o contêiner que inclui o SWF é um arquivo HTML; um arquivo SWF é usado, com menos frequência, em toda ou parte da interface de usuário de um aplicativo de área de trabalho.

À medida que você trabalha em aplicativos mais avançados, talvez precise configurar a comunicação entre o arquivo SWF e o aplicativo de contêiner. Por exemplo, uma página da Web normalmente exibe texto ou outras informações em HTML e inclui um arquivo SWF para exibir conteúdo visual dinâmico, como um gráfico ou vídeo. Nesse caso, talvez seja útil especificar que, quando os usuários clicam em um botão na página da Web, alguma coisa deve mudar no arquivo SWF. O ActionScript contém um mecanismo, conhecido como API externa, que facilita esse tipo de comunicação entre o ActionScript em um arquivo SWF e outro código no aplicativo de contêiner.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes relevantes a este recurso:

Aplicativo de contêiner O aplicativo no qual o Flash Player está executando o arquivo SWF, como um navegador da Web e a página HTML que inclui o conteúdo do Flash Player ou um aplicativo do AIR que carregue o SWF em uma página da Web.

Projetor Um arquivo executável que inclui conteúdo SWF e uma versão incorporada do Flash Player. Você pode criar um arquivo de projetor usando o Flash Professional ou o Flash Player independente. Os projetores normalmente são usados para distribuir arquivos SWF por CD-ROM ou em situações similares onde o tamanho do download não é um problema e o autor do SWF quer garantir que o usuário possa executar o arquivo SWF, independentemente da instalação do Flash Player no computador do usuário.

Proxy Um aplicativo ou código mediador que chama o código de um aplicativo (o “aplicativo externo”) em nome de outro aplicativo (o “aplicativo de chamada”) e retorna valores para o aplicativo de chamada. Um proxy pode ser usado por vários motivos, como:

- Para simplificar o processo de chamada de funções externas, convertendo as chamadas de funções nativas do aplicativo de chamada no formato reconhecido pelo aplicativo externo.
- Para obter uma solução alternativa para segurança ou outras restrições que impedem a comunicação direta do chamador com o aplicativo externo.

Serializar Converter valores de objetos ou dados em um formato que pode ser usado para transmitir os valores em mensagens entre dois sistemas de programação, como via Internet ou entre dois aplicativos diferentes em execução em um único computador.

Trabalho por meio de exemplos

Muitos exemplos de código fornecidos são pequenas listagens apenas para demonstração, não exemplos de trabalho completos ou código que verifica valores. Como o uso da API externa requer (por definição) a gravação do código do ActionScript, bem como do código de um aplicativo de contêiner, o teste dos exemplos envolve a criação de um contêiner (por exemplo, uma página da Web que contém o arquivo SWF) e o uso das listagens de código para interagir com o contêiner.

Para testar um exemplo de comunicação entre ActionScript e JavaScript:

1. Crie um novo documento usando o Flash Professional e salve-o em seu computador.
2. No menu principal, escolha Arquivo > Configurações de publicação.
3. Na caixa de diálogo Configurações de publicação, na aba Formatos, verifique se as caixas de seleção Flash e HTML estão marcadas.
4. Clique no botão Publicar. Isso gera um arquivo SWF e um arquivo HTML na mesma pasta, com o mesmo nome usado para salvar o documento. Clique em OK para fechar a caixa de diálogo Configurações de publicação.
5. Desmarque a caixa de seleção HTML. Agora que a página HTML foi gerada, você irá modificá-la para adicionar o código do JavaScript apropriado. Desmarcar a caixa de seleção HTML garante que, após a modificação da página HTML, o Flash não substituirá as alterações por uma nova página HTML ao publicar o arquivo SWF.

6 Clique em OK para fechar a caixa de diálogo Configurações de publicação.

7 Com um aplicativo de editor de texto ou HTML, abra o arquivo HTML criado pelo Flash ao publicar o arquivo SWF. No código-fonte HTML, adicione as tags `script` de abertura e fechamento e copie-as no código do JavaScript na listagem de código de exemplo:

```
<script>
// add the sample JavaScript code here
</script>
```

8 Salve o arquivo HTML e volte ao Flash.

9 Selecione o quadro-chave no Quadro 1 da Linha de tempo e abra o painel Ações.

10 Copie a listagem de código do ActionScript no painel Script.

11 No menu principal, escolha Arquivo > Publicar para atualizar o arquivo SWF com as alterações feitas.

12 Usando um navegador da Web, abra a página HTML editada para visualizá-la e testar a comunicação entre o ActionScript e a página HTML.

Para testar um exemplo de comunicação de contêiner entre ActionScript e ActiveX:

1 Crie um novo documento usando o Flash Professional e salve-o em seu computador. Salve-o na pasta onde o aplicativo de contêiner deverá localizar o arquivo SWF.

2 No menu principal, escolha Arquivo > Configurações de publicação.

3 Na caixa de diálogo Configurações de publicação, na aba Formatos, verifique se apenas a caixa de seleção Flash está marcada.

4 No campo Arquivo próximo à caixa de seleção Flash, clique no ícone de pasta para selecionar a pasta na qual o arquivo SWF será publicado. Uma vez definido o local do arquivo SWF, você pode, por exemplo, manter o documento em uma determinada pasta e colocar o arquivo SWF publicado em outra pasta, como a que contém o código-fonte do aplicativo de contêiner.

5 Selecione o quadro-chave no Quadro 1 da Linha de tempo e abra o painel Ações.

6 Copie o código do ActionScript do exemplo no painel Script.

7 No menu principal, escolha Arquivo > Publicar para republicar o arquivo SWF.

8 Crie e execute o aplicativo de contêiner para testar a comunicação entre o ActionScript e o aplicativo de contêiner.

Para ver exemplos completos da utilização da API externa para se comunicar com uma página HTML, consulte o seguinte tópico:

- "Exemplo de API externa: comunicação entre ActionScript e JavaScript em um navegador da Web" na página 877

Estes exemplos incluem o código completo, inclusive o código de verificação de erro do ActionScript e do contêiner, que deve ser usado ao programar o código com a API externa. Para obter outro exemplo completo de uso da API externa, consulte o exemplo da classe ExternalInterface na Referência do ActionScript 3.0.

Requisitos e vantagens da API externa

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A API externa é a parte do ActionScript que fornece um mecanismo para comunicação entre o ActionScript e o código em execução em um "aplicativo externo" que está agindo como contêiner para o Flash Player (normalmente um navegador da Web ou aplicativo de projetor dedicado). No ActionScript 3.0, a funcionalidade da API externa é fornecida pela classe ExternalInterface. Nas versões do Flash Player anteriores ao Flash Player 8, a ação `fscommand()` era usada para estabelecer a comunicação com o aplicativo de contêiner. A classe ExternalInterface é a substituição para `fscommand()`.

Nota: *Se precisar usar a função `fscommand()` antiga (por exemplo, para manter a compatibilidade com aplicativos mais antigos ou para interagir com um aplicativo de contêiner SWF de terceiros ou o Flash Player), ela ainda está disponível como uma função no pacote flash.system.*

A classe ExternalInterface é um subsistema que permite a comunicação fácil entre o ActionScript e o Flash Player com JavaScript em uma página HTML.

A classe ExternalInterface só está disponível nas seguintes condições:

- Em todas as versões compatíveis do Internet Explorer para Windows (5.0 e posterior)
- Em qualquer navegador compatível com a interface NPRuntime, que no momento inclui o Firefox 1.0 e posterior, o Mozilla 1.7.5 e posterior, o Netscape 8.0 e posterior e o Safari 1.3 e posterior.
- Em um aplicativo do AIR onde o SWF está incorporado em uma página HTML exibida pelo controle HTMLLoader.

Em todas as outras situações (como a execução em um player dedicado), a propriedade `ExternalInterface.available` retorna `false`.

No ActionScript, é possível chamar uma função JavaScript na página HTML. A API externa tem uma funcionalidade aprimorada em comparação com `fscommand()`:

- É possível usar qualquer função JavaScript, não apenas as funções que podem ser usadas com a função `fscommand()`.
- É possível transmitir qualquer número de argumentos, com qualquer nome; você não fica limitado a transmitir um comando e um único argumento de string. Desse modo, a API externa tem muito mais flexibilidade do que `fscommand()`.
- É possível transmitir vários tipos de dados (como booliano, número e string); você não está limitado aos parâmetros String.
- Você pode receber o valor de uma chamada e esse valor retorna imediatamente para o ActionScript (como o valor de retorno da chamada feita).

Importante: *Se o nome dado à ocorrência do Flash Player em uma página HTML (atributo `id` da tag `object`) incluir um hífen (`-`) ou outros caracteres que são definidos como operadores no JavaScript (como `+`, `*`, `/`, `\`, `.`, e assim por diante), as chamadas ExternalInterface do ActionScript não vão funcionar quando a página da Web do contêiner é visualizada no Internet Explorer. Além disso, se as tags HTML que definem a ocorrência do Flash Player (as tags `object` e `embed`) estiverem em uma tag HTML `form`, as chamadas ExternalInterface do ActionScript não vão funcionar.*

Uso da classe ExternalInterface

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A comunicação entre o ActionScript e o aplicativo de contêiner pode ser realizada de duas formas: o ActionScript pode chamar o código (como uma função do JavaScript) definido no contêiner ou o código do contêiner pode chamar uma função do ActionScript que foi designada como chamável. De qualquer modo, as informações podem ser enviadas para o código que está sendo chamado e os resultados podem ser retornados para o código que está fazendo a chamada.

Para facilitar essa comunicação, a classe ExternalInterface inclui duas propriedades estáticas e dois métodos estáticos. Esses métodos e propriedades são usados para obter informações sobre a conexão da interface externa, para executar o código no contêiner do ActionScript e para disponibilizar as funções do ActionScript para serem chamadas pelo contêiner.

Obtenção de informações sobre o contêiner externo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A propriedade `ExternalInterface.available` indica se o Flash Player atual está em um contêiner que oferece uma interface externa. Se a interface externa estiver disponível, esta propriedade será `true`; caso contrário, ela será `false`. Antes de usar qualquer outra funcionalidade na classe ExternalInterface, sempre verifique se o contêiner atual oferece suporte à comunicação da interface externa do seguinte modo:

```
if (ExternalInterface.available)
{
    // Perform ExternalInterface method calls here.
}
```

Nota: *A propriedade `ExternalInterface.available` informa se o contêiner atual é um tipo compatível com a conectividade de ExternalInterface. Ela não informa se o JavaScript está ativado no navegador atual.*

A propriedade `ExternalInterface.objectID` permite determinar o identificador exclusivo da ocorrência do Flash Player (especificamente, o atributo `id` da tag `object` no Internet Explorer ou o atributo `name` da tag `embed` nos navegadores que usam a interface NPRuntime). Essa ID exclusiva representa o documenta SWF atual no navegador e pode ser usada para fazer referência ao documento SWF (por exemplo, ao chamar uma função do JavaScript em uma página HTML de contêiner). Quando o contêiner do Flash Player não é um navegador da Web, essa propriedade é `null`.

Chamada do código externo a partir do ActionScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `ExternalInterface.call()` executa o código no aplicativo de contêiner. Ele requer pelo menos um parâmetro, uma string que contém o nome da função a ser chamada no aplicativo de contêiner. Quaisquer parâmetros adicionais transmitidos para o método `ExternalInterface.call()` são transmitidos ao longo do contêiner como parâmetros da chamada de função.

```
// 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);
```

Se o contêiner for uma página HTML, esse método invocará a função do JavaScript com o nome especificado, que deve ser definido em um elemento `script` na página HTML de contêiner. O valor de retorno da função do JavaScript é transmitido novamente para o ActionScript.

```
<script language="JavaScript">
    // adds two numbers, and sends the result back to ActionScript
    function addNumbers(num1, num2)
    {
        return (num1 + num2);
    }
</script>
```

Se o contêiner for algum outro contêiner ActiveX, esse método fará com que o controle ActiveX do Flash Player envie o evento `FlashCall`. O nome da função e os parâmetros especificados são serializados em uma string XML pelo Flash Player. O contêiner pode acessar essas informações na propriedade `request` do objeto de evento e usá-las para determinar como seu próprio código deve ser executado. Para retornar um valor para o ActionScript, o código do contêiner chama o método `SetReturnValue()` do objeto ActiveX, transmitindo o resultado (serializado em uma string XML) como um parâmetro desse método. Para obter mais informações sobre o formato XML usado para essa comunicação, consulte "O formato XML da API externa" na página 875.

Independentemente de o contêiner ser um navegador da Web ou outro contêiner ActiveX, se a chamada falhar ou o método do contêiner não especificar um valor de retorno, `null` será retornado. O método `ExternalInterface.call()` lançará uma exceção SecurityError se o ambiente incluído pertencer a uma caixa de proteção de segurança para a qual o código de chamada não tem acesso. Para solucionar isso temporariamente, defina um valor apropriado para `allowScriptAccess` no ambiente incluído. Por exemplo, para alterar o valor de `allowScriptAccess` em uma página HTML, edite o atributo adequado nas tags `object` e `embed`.

Chamada do código do ActionScript a partir do contêiner

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um contêiner só pode chamar o código do ActionScript que esteja em uma função - nenhum outro código do ActionScript pode ser chamado por um contêiner. Para chamar um função do ActionScript a partir do aplicativo de contêiner, você deve fazer duas coisas: registrar a função com a classe ExternalInterface e chamá-la a partir do código do contêiner.

Primeiro, você deve registrar a função do ActionScript para indicar se ela deve ser disponibilizada para o contêiner. Use o método `ExternalInterface.addCallback()` do seguinte modo:

```
function callMe(name:String):String
{
    return "busy signal";
}
ExternalInterface.addCallback("myFunction", callMe);
```

O método `addCallback()` tem dois parâmetros. O primeiro, um nome de função como String, é o nome pelo qual a função será reconhecida pelo contêiner. O segundo parâmetro é a função real do ActionScript que será executada quando o contêiner chamar o nome de função definido. Como esses nomes são diferentes, você pode especificar um nome de função que será usado pelo contêiner, mesmo se a função real do ActionScript tiver um nome diferente. Isso é especialmente útil se o nome da função não for conhecido - por exemplo, se uma função anônima for especificada ou se a função a ser chamada for determinada em tempo de execução.

Depois que uma função do ActionScript é registrada com a classe ExternalInterface, o contêiner pode realmente chamar a função. O modo como isso é feito varia de acordo com o tipo de contêiner. Por exemplo, no código do JavaScript em um navegador da Web, a função do ActionScript é chamada com o nome de função registrado, visto que este é um método do objeto de navegador do Flash Player (isto é, um método do objeto JavaScript que representa a tag `object` ou `embed`). Em outras palavras, os parâmetros são transmitidos e um resultado é retornado como uma função local que está sendo chamada.

```
<script language="JavaScript">
    // callResult gets the value "busy signal"
    var callResult = flashObject.myFunction("my name");
</script>
...
<object id="flashObject"...>
    ...
    <embed name="flashObject".../>
</object>
```

Como alternativa, ao chamar uma função do ActionScript em um arquivo SWF em execução em um aplicativo de área de trabalho, o nome de função registrado e os parâmetros devem ser serializados em uma string XML. Em seguida, a chamada é realmente feita para o método `CallFunction()` do controle ActiveX com a string XML como parâmetro. Para obter mais informações sobre o formato XML usado para essa comunicação, consulte "O formato XML da API externa" na página 875.

De qualquer modo, o valor de retorno da função do ActionScript é transmitido novamente para o código do contêiner, diretamente como um valor quando o chamador é o código JavaScript em um navegador ou serializado com uma string XML quando o chamador é um contêiner ActiveX.

O formato XML da API externa

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A comunicação entre o ActionScript e um aplicativo que hospeda o controle ActiveX do Shockwave Flash usa um formato XML específico para codificar chamadas e valores de função. Duas partes do formato XML são usadas pela API externa. Um formato é usado para representar chamadas de função. Outro formato é usado para representar valores individuais; esse formato é usado para parâmetros em funções e em valores de retorno de função. O formato XML das chamadas de função é usado para chamadas feitas e recebidas pelo ActionScript. Para uma chamada de função a partir do ActionScript, o Flash Player transmite o XML para o contêiner; para uma chamada a partir do contêiner, o Flash Player espera que o aplicativo de contêiner o transmita como uma string XML nesse formato. O fragmento XML a seguir mostra o exemplo de uma chamada de função em formato XML:

```
<invoke name="functionName" returntype="xml">
    <arguments>
        ... (individual argument values)
    </arguments>
</invoke>
```

O nó raiz é o nó `invoke`. Ele tem dois atributos: `name` indica o nome da função a ser chamada e `returntype` é sempre `xml`. Se a chamada de função incluir parâmetros, o nó `invoke` terá um nó filho `arguments`, cujos nós filho serão os valores de parâmetro formatados que usam o formato de valor individual explicado a seguir.

Os valores individuais, incluindo parâmetros de função e valores de retorno de função, usam um esquema de formatação que inclui informações sobre o tipo de dados além dos valores reais. A tabela a seguir lista as classes do ActionScript e o formato XML usado para codificar valores desse tipo de dados:

Classe/valor do ActionScript	Classe/valor do C#	Formato	Comentários
null	null	<null/>	
Booliano `true`	bool true	<true/>	
Booliano `false`	bool false	<false/>	
String	string	<string>valor da string</string>	
Número, int, uint	único, duplo, int, uint	`<number>27.5</number>` `<number>-12</number>`	

Array (os elementos podem ser de tipos diferentes)	Uma coleção que permite elementos de tipos diferentes, como ArrayList ou object[]	`<array>`
		`<property id="0">`
		`<number>27.5</number>`
		`</property>`
		`<property id="1">`
		`<string>Hello there!</string>`
		`</property>`
		`...`
		`</array>`
			O nó `property` define elementos individuais e o atributo `id` é o índice numérico baseado em zero.

Objeto	Um dicionário com chaves de string e valores de objeto, como HashTable com chaves de string	`<object>`
		`<property id="name">`
		`<string>John Doe</string>`
		`</property>`
		`<property id="age">`
		`<string>33</string>`
		`</property>`
		`...`
		`</object>`
			O nó `property` define propriedades individuais e o atributo `id` é o nome da propriedade (uma string).

Outras classes internas ou personalizadas		`<null/> or` `<object></object>`	O ActionScript codifica outros objetos como nulos ou como um objeto vazio. De qualquer modo, os valores de propriedade são perdidos.

Nota: *Por exemplo, esta tabela mostra as classes C# equivalentes além das classes do ActionScript; no entanto, a API externa pode ser usada para comunicação com qualquer linguagem de programação ou tempo de execução compatível com os controles ActiveX, não se limitando aos aplicativos C#.*

Quando está criando seus próprios aplicativos usando a API externa com um aplicativo de contêiner ActiveX, você provavelmente achará útil gravar um proxy que irá converter as chamadas de função nativas no formato XML serializado. Para obter um exemplo de uma classe de proxy gravada em C#, consulte Dentro da classe ExternalInterfaceProxy.

Exemplo de API externa: comunicação entre ActionScript e JavaScript em um navegador da Web

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Este aplicativo de amostra demonstra técnicas adequadas de comunicação entre o ActionScript e o JavaScript em um navegador da Web, no âmbito de um aplicativo de mensagem instantânea que permite que uma pessoa converse consigo mesma (daí o nome do aplicativo: Introvert IM). As mensagens são enviadas entre um formulário HTML na página da Web e uma interface SWF usando a API externa. As técnicas demonstradas neste exemplo incluem as seguintes:

- Início adequado da comunicação, verificando se o navegador está preparado para se comunicar antes de estabelecer uma comunicação
- Verificação da compatibilidade entre o contêiner e a API externa
- Chamada das funções do JavaScript a partir do ActionScript, transmitindo parâmetros e recebendo valores em resposta
- Disponibilização dos métodos do ActionScript para serem chamados pelo JavaScript, e realização dessas chamadas

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Introvert IM estão localizados na pasta Amostras/IntrovertIM_HTML. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
IntrovertIMApp.fla ou IntrovertIMApp.mxml	O arquivo de aplicativo principal no Flash (FLA) ou Flex (MXML).
com/example/programmingas3/introvertIM/IMManager.as	A classe que estabelece e gerencia a comunicação entre o ActionScript e o contêiner.
com/example/programmingas3/introvertIM/IMMessageEvent.as	Tipo de evento personalizado, enviado pela classe IMManager quando uma mensagem é recebida do contêiner.
com/example/programmingas3/introvertIM/IMStatus.as	Enumeração cujos valores representam diferentes status de "disponibilidade" que podem ser selecionados no aplicativo.
html-flash/IntrovertIMApp.html ou html-template/index.template.html	A página HTML do aplicativo para Flash (html-flash/IntrovertIMApp.html) ou o modelo usado para criar a página HTML do contêiner do aplicativo para Adobe Flex (html-template/index.template.html). Esse arquivo contém todas as funções do JavaScript que fazem parte do contêiner do aplicativo.

Preparação da comunicação entre o ActionScript e o navegador

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A API externa normalmente é usada para permitir a comunicação dos aplicativos do ActionScript com um navegador da Web. Usando a API externa, os métodos do ActionScript podem chamar o código gravado em JavaScript e vice-versa. Devido à complexidade dos navegadores e ao modo de renderização interna das páginas, não é possível garantir que um documento SWF registrará seus retornos de chamada antes da execução do primeiro JavaScript na página HTML. Por esse motivo, antes de chamar funções no documento SWF a partir do JavaScript, o documento SWF sempre deve chamar a página HTML para notificar se o documento SWF está pronto para aceitar conexões.

Por exemplo, por meio de uma série de etapas realizadas pela classe IMManager, o Introvert IM determina se o navegador está pronto para comunicação e prepara o arquivo SWF para comunicação. A primeira etapa, determinar quando o navegador está pronto para comunicação, acontece no construtor IMManager do seguinte modo:

```
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.");
    }
}
```

Primeiro, o código verifica se a API externa está realmente disponível no contêiner atual usando a propriedade `ExternalInterface.available`. Se estiver, o processo de configuração da comunicação é iniciado. Como exceções de segurança e outros erros podem ocorrer ao tentar se comunicar com um aplicativo externo, o código fica entre um bloco `try` (os blocos `catch` correspondentes foram omitidos da listagem para facilitar).

O código a seguir chama o método `isContainerReady()`, listado aqui:

```
private function isContainerReady():Boolean
{
    var result:Boolean = ExternalInterface.call("isReady");
    return result;
}
```

O método `isContainerReady()`, por sua vez, usa o método `ExternalInterface.call()` para chamar a função `isReady()` do JavaScript do seguinte modo:

```
<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>
```

A função `isReady()` simplesmente retorna o valor da variável `jsReady`. Essa variável é inicialmente `false`; assim que o evento `onload` da página da Web é acionado, o valor da variável muda para `true`. Em outras palavras, se o ActionScript chamar a função `isReady()` antes que a página seja carregada, o JavaScript retornará `false` para `ExternalInterface.call("isReady")` e, consequentemente, o método `isContainerReady()` do ActionScript retornará `false`. Assim que a página for carregada, a função `isReady()` do JavaScript retornará `true`, de modo que o método `isContainerReady()` do ActionScript também retornará `true`.

De volta ao construtor IMManager, uma de duas coisas acontecem dependendo da disponibilidade do contêiner. Se `isContainerReady()` retornar `true`, o código simplesmente chamará o método `setupCallbacks()`, que concluirá o processo de configuração da comunicação com o JavaScript. Por outro lado, se `isContainerReady()` retornar `false`, o processo será basicamente suspenso. Um objeto Timer é criado e suspenso para chamar o método `timerHandler()` a cada 100 milissegundos do seguinte modo:

```
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();
    }
}
```

Sempre que é chamado, o método `timerHandler()` verifica mais uma vez o resultado do método `isContainerReady()`. Assim que o contêiner é inicializado, esse método retorna `true`. Em seguida, o código para o objeto Timer e chama o método `setupCallbacks()` para finalizar o processo de configuração da comunicação com o navegador.

Exposição dos métodos do ActionScript ao JavaScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Conforme mostrou o exemplo anterior, assim que o código determina que o navegador está pronto, o método `setupCallbacks()` é chamado. Esse método prepara o ActionScript para receber chamadas do JavaScript, conforme mostrado a seguir:

```
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");
}
```

O método `setCallBacks()` termina a tarefa de preparação de comunicação com o contêiner chamando `ExternalInterface.addCallback()` para registrar os dois métodos que estarão disponíveis para serem chamados a partir do JavaScript. Neste código, o primeiro parâmetro - o nome pelo qual o método é conhecido pelo JavaScript (`"newMessage"` e `"getStatus"`) - é igual ao nome do método no ActionScript. Neste caso, não seria útil usar nomes diferentes, de modo que o mesmo nome foi reutilizado para simplificar. Finalmente, o método `ExternalInterface.call()` é usado para chamar a função `setSWFIsReady()` do JavaScript, que avisa ao contêiner que as funções do ActionScript foram registradas.

Comunicação do ActionScript com o navegador

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O aplicativo Introvert IM demonstra diversos exemplos de chamada das funções do JavaScript na página do contêiner. No caso mais simples (um exemplo do método `setupCallbacks()`), a função `setSWFIsReady()` do JavaScript é chamada sem transmitir nenhum parâmetro nem receber um valor de retorno:

```
ExternalInterface.call("setSWFIsReady");
```

Em outro exemplo do método `isContainerReady()`, o ActionScript chama a função `isReady()` e recebe um valor booliano em resposta:

```
var result:Boolean = ExternalInterface.call("isReady");
```

Também é possível transmitir parâmetros para as funções do JavaScript usando a API externa. Por exemplo, considere o método `sendMessage()` da classe IMManager, que é chamado quando o usuário está enviando uma nova mensagem para seu "parceiro de conversa":

```
public function sendMessage(message:String):void
{
    ExternalInterface.call("newMessage", message);
}
```

Novamente, `ExternalInterface.call()` é usado para chamar a função designada do JavaScript, notificando o navegador sobre a nova mensagem. Além disso, a mensagem propriamente dita é transmitida como um parâmetro adicional para `ExternalInterface.call()` e, consequentemente, transmitida como parâmetro para a função `newMessage()` do JavaScript.

Chamada do código do ActionScript a partir do JavaScript

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A comunicação é uma rua de mão dupla e o aplicativo Introvert IM não foge à regra. O cliente IM do Flash Player não só chama o JavaScript para enviar mensagens, mas o formulário HTML chama o código do JavaScript para enviar mensagens e solicitar informações do arquivo SWF também. Por exemplo, quando o arquivo SWF avisa o contêiner que terminou de estabelecer o contato e está pronto para se comunicar, a primeira coisa que o navegador faz é chamar o método `getStatus()` da classe IMManager para recuperar o status de disponibilidade inicial do usuário a partir do cliente IM do SWF. Isso é feito na página da Web, na função `updateStatus()`, do seguinte modo:

```
<script language="JavaScript">
...
function updateStatus()
{
    if (swfReady)
    {
        var currentStatus = getSWF("IntrovertIMApp").getStatus();
        document.forms["imForm"].status.value = currentStatus;
    }
}
...
</script>
```

O código verifica o valor da variável `swfReady`, que controla se o arquivo SWF notificou o navegador sobre o registro dos métodos com a classe ExternalInterface. Se o arquivo SWF estiver pronto para receber comunicação, a próxima linha (`var currentStatus =` ...) realmente chamará o método `getStatus()` na classe IMManager. Três coisas acontecem nesta linha de código:

- A função `getSWF()` do JavaScript é chamada, retornando uma referência ao objeto do JavaScript que representa o arquivo SWF. O parâmetro transmitido para `getSWF()` determina qual objeto de navegador é retornado caso haja mais de um arquivo SWF em uma página HTML. O valor transmitido para esse parâmetro deve corresponder ao atributo `id` da tag `object` e ao atributo `name` da tag `embed` usados para incluir o arquivo SWF.
- Usando a referência ao arquivo SWF, o método `getStatus()` é chamado embora seja um método do objeto SWF. Nesse caso, o nome da função "`getStatus`" é usado porque esse é o nome com o qual a função do ActionScript foi registrada com `ExternalInterface.addCallback()`.
- O método `getStatus()` do ActionScript retorna um valor e esse valor é atribuído à variável `currentStatus`, que é atribuída como conteúdo (a propriedade `value`) do campo de texto `status`.

Nota: *Se estiver seguindo o código, você provavelmente percebeu que, no código-fonte da função `updateStatus()`, a linha de código que chama a função `getSWF()` é realmente escrita do seguinte modo: var currentStatus = getSWF("${application}").getStatus(). O texto `${application}` é um alocador de espaço contido no modelo da página HTML. Quando o Adobe Flash Builder gera a página HTML real para o aplicativo, esse alocador é substituído pelo mesmo texto usado como o atributo `id` da tag `object` e o atributo `name` da tag `embed` (que corresponde a `IntrovertIMApp` no exemplo). Este é o valor esperado pela função `getSWF()`.*

A função `sendMessage()` do JavaScript demonstra a transmissão de um parâmetro como uma função do ActionScript. (`sendMessage()` é afunção que foi chamada quando o usuário pressionou o botão Enviar na página HTML.

```
<script language="JavaScript">
...
function sendMessage(message)
{
    if (swfReady)
    {
        ...
        getSWF("IntrovertIMApp").newMessage(message);
    }
}
...
</script>
```

O método `newMessage()` do ActionScript espera um parâmetro, de modo que a variável `message` do JavaScript é transmitida para o ActionScript como um parâmetro na chamada do método `newMessage()` no código do JavaScript.

Detecção do tipo de navegador

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Devido às diferenças de acesso ao conteúdo dos navegadores, é importante sempre usar o JavaScript para detectar qual navegados está sendo executado pelo usuário e para acessar o filme de acordo com a sintaxe específica do navegador, usando o objeto de janela ou documento, como mostra a função `getSWF()` do JavaScript neste exemplo:

```
<script language="JavaScript">
...
function getSWF(movieName)
{
    if (navigator.appName.indexOf("Microsoft") != -1)
    {
        return window[movieName];
    }
    else
    {
        return document[movieName];
    }
}
...
</script>
```

Se o seu script não detectar o tipo de navegador do usuário, o usuário poderá observar um comportamento inesperado ao reproduzir arquivos SWF em um contêiner HTML.

Capítulo 48: Validação da assinatura XML no AIR

Adobe AIR 1.5 e posterior

Use as classes da API do XMLSignatureValidator no Adobe® AIR® para validar assinaturas digitais em conformidade com um subconjunto de recomendações do W3C para processamento e sintaxe de assinatura XML (http://http://www.w3.org/TR/xmldsig-core/). As assinaturas XML podem ser usadas para verificar a integridade e a identidade de quem assina dados ou informações.

Assinaturas XML podem ser usadas para validar mensagens ou recursos baixados por seu aplicativo. Por exemplo, se seu aplicativo fornece serviços mediante assinatura, você pode encapsular os termos da assinatura em um documento XML assinado. Se alguém tentar alterar o documento de assinatura, a validação falhará.

Você também pode usar um assinatura XML para ajudar a validar recursos baixados usados por seu aplicativo, incluindo um manifesto assinado contendo compilações desses recursos. Seu aplicativo pode verificar se os recursos não foram alterados, comparando a compilação no arquivo assinado a uma compilação computada dos bytes carregados. Isso é especialmente valioso quando o recurso baixado é um arquivo SWF ou outro conteúdo ativo que você deseje executar na caixa de proteção de segurança do aplicativo.

Conceitos básicos sobre validação da assinatura XML

Adobe AIR 1.5 e posterior

Veja uma explicação rápida e exemplos de códigos para a validação de assinaturas de XML nos seguintes artigos de início rápido do Adobe Developer Connection:

- Criação e validação de assinaturas XML (Flex)
- Criação e validação de assinaturas XML (Flash)

O Adobe® AIR® fornece a classe XMLSignatureValidator e a interface IURIDreferencer para validar assinaturas XML. A sintaxe XML aceita pela classe XMLSignatureValidator é um subconjunto das recomendações do W3C para processamento e sintaxe de assinatura XML. (Como somente um subconjunto da recomendação tem suporte, nem todas as assinaturas permitidas podem ser validadas). O AIR não fornece uma API para a criação de assinaturas XML.

Classes de validação da assinatura XML

Adobe AIR 1.5 e posterior

A API de validação da assinatura XML inclui estas classes:

Pacote	Classes
flash.security	• XMLSignatureValidator • IURIDereferencer (interface) Constantes de string de XMLSignatureValidator são definidas nas seguintes classes: • ReferencesValidationSetting • RevocationCheckSettings • SignatureStatus • SignerTrustSettings
flash.events	• Evento • ErrorEvent

Uso das classes de validação da assinatura XML

Adobe AIR 1.5 e posterior

Para usar a classe XMLSignatureValidator para validar uma assinatura XML, você deve:

- Criar um objeto XMLSignatureValidator
- Fornecer uma implementação da interface IURIDereferencer. O objeto XMLSignatureValidator chama o método `dereference()` de IURIDereferencer, enviando a URI para cada referência em uma assinatura. O método `dereference()` deve resolver a URI e retornar os dados referenciados (que devem estar no mesmo documento como a assinatura, ou podem estar em um recurso externo).
- Defina as configurações de confiança do certificado, verificação de revogação e validação de referência do objeto XMLSignatureValidator conforme apropriado para o seu aplicativo.
- Adicione ouvintes do evento para os eventos `complete` e `error`.
- Chame o método `verify`, enviando a assinatura para verificação.
- Manipule os eventos `complete` e `error` e interprete os resultados.

O exemplo a seguir implementa uma função `validate()` que verifica a validade de uma assinatura XML. As propriedades de XMLSignatureValidator são definidas de forma que o certificado de assinatura deve ficar na armazenagem confiável do sistema, ou encadeado a ela. O exemplo também supõe que uma classe IURIDereferencer adequada denominada *XMLDereferencer* exista.

```
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);
}
```

O processo de validação da assinatura XML

Adobe AIR 1.5 e posterior

Quando você chama o método `verify()` de XMLSignatureValidator, o AIR executa estas etapas:

- O tempo de execução verifica a integridade criptográfica da assinatura usando a chave pública do certificado de assinatura.
- O tempo de execução estabelece a integridade criptográfica, a identidade e a confiabilidade do certificado com base nas configurações atuais do objeto XMLSignatureValidator.

  A confiança inserida no certificado de assinatura é a chave para a integridade do processo de validação. A validação da assinatura é conduzida com um processo criptográfico bem definido, mas a confiabilidade do certificado de assinatura é um julgamento que não pode ser realizado de forma algorítmica.

  Em geral, há três formas de decidir se um certificado é confiável:

  - Confiando em autoridades de certificação e no armazenamento confiável do sistema operacional.

  - Obtendo, diretamente de quem assina, uma cópia do certificado, outro certificado que sirva como âncora confiável do certificado ou informações suficientes para identificar confiavelmente o certificado, como a chave pública.
  - Perguntando ao usuário final do aplicativo se ele confia no certificado. Tal pergunta é inválida com certificados autoassinados, pois as informações de identificação no certificado não são confiáveis por inerência.
- O tempo de execução verifica a integridade criptográfica dos dados assinados.

  Os dados assinados são verificados com a ajuda da implementação do seu IURIDereferencer. Para cada referência no documento de assinatura, o método `dereference()` de implementação do IURIDereferencer é chamado. Os dados retornados pelo método `dereference()` são usados para computar a compilação de referência. Esse valor de compilação é comparado à compilação registrada no documento de assinatura. Se a compilação for correspondente, os dados ainda não foram alterados desde a assinatura.

  Uma consideração importante ao contar com os resultados de validação de assinatura XML é que apenas o que é assinado é seguro. Por exemplo, considere um manifesto assinado listando os arquivos de um pacote. Quando o XMLSignatureValidator verifica a assinatura, ele verifica somente se o manifesto não foi alterado. Os dados dos arquivos não são assinados, portanto a assinatura ainda será válida quando arquivos referenciados no manifesto forem alterados ou excluídos.

  Nota: *Para verificar arquivos em tal manifesto, você pode computar a compilação dos dados do arquivo (usando o mesmo algoritmo de hash usado no manifesto) e comparar o resultado com a compilação armazenada no manifesto assinado. Em alguns casos, e necessário verificar também a presença de arquivos adicionais.*

Interpretação dos resultados de validação

Adobe AIR 1.5 e posterior

Os resultados de validação são relatados pelas propriedades de status do objeto XMLSignatureValidator. Essas propriedades podem ser lidas depois que o objeto validador despacha o evento *complete*. As quatro propriedades de status incluem: `validityStatus`, `digestStatus`, `identityStatus` e `referencesStatus`.

A propriedade validityStatus

Adobe AIR 1.5 e posterior

A propriedade `validityStatus` relata a validade geral da assinatura. A `validityStatus` depende do estado de outras três propriedades de status e pode ter um destes valores:

- `valid` — Se `digestStatus`, `identityStatus` e `referencesStatus` forem todas `válidas`.
- `invalid` — Se uma ou mais propriedades de status individuais forem `inválidas`.
- `unknown` — se uma ou mais propriedades de status individual forem `desconhecidas` e nenhum status individual for `inválido`.

A propriedade digestStatus

Adobe AIR 1.5 e posterior

A propriedade `digestStatus` relata os resultados da verificação criptográfica da compilação da mensagem. Esta propriedade `digestStatus` pode ter um dos seguintes valores:

- `valid` — Se o documento de assinatura se mantiver sem alterações desde que a assinatura tenha sido realizada.
- `invalid` — se o documento de assinatura foi alterado ou foi malformado.

- `unknown` — Se o método `verify()` não tiver sido concluído sem erros.

A propriedade identityStatus

Adobe AIR 1.5 e posterior

A propriedade `identityStatus` relata o status do certificado de assinatura. O valor dessa propriedade depende de vários fatores, incluindo:

- a integridade criptográfica do certificado
- se o certificado foi expirado ou revogado
- se o certificado é confiável na máquina atual
- o estado do objeto XMLSignatureValidator (como se outros certificados foram adicionados para criar a cadeia confiável, se esses certificados são confiáveis e os valores das propriedades `useSystemTrustStore` e `revocationCheckSettings`)

A propriedade `identityStatus` pode ter estes valores:

- `valid` — Para ser considerado válido, o certificado de assinatura deve atender estas condições:
  - O certificado de assinatura deve estar inalterado.
  - O certificado de assinatura não deve ter expirado ou sido revogado, exceto quando um carimbo de data/hora válido estiver presente na assinatura. Se a assinatura tiver um carimbo de data/hora, o certificado será considerado válido, desde que estivesse válido no momento da assinatura do documento. (O certificado usado pelo serviço de carimbo de data/hora para assinar o carimbo de data/hora deve se vincular a um certificado de raiz confiável no computador do usuário).
  - O certificado de assinatura é confiável. Um certificado é confiável se estiver na armazenagem confiável do sistema ou encadeado com outro certificado na armazenagem confiável do sistema e se você definir a propriedade `useSystemTrustStore` como true. Você também pode designar um certificado como confiável usando o método `addCertificate()` do objeto XMLSignatureValidator.
  - O certificado é, na realidade, o certificado de assinatura.
- `invalid` — O certificado está expirado ou revogado e nenhum carimbo de data e hora fornecendo validade de horário da assinatura está presente, ou o certificado foi alterado.
- `unknown` — Se o certificado não for inválido, mas não for confiável. Certificados autoassinados, por exemplo, serão reportados como `unknown` (a menos que explicitamente confiáveis). O `identityStatus` também é reportado como `unknown` se o método `verify()` não tiver sido concluído sem erros ou se a identidade não tiver sido verificada porque a compilação de assinatura é inválida.

A propriedade referencesStatus

Adobe AIR 1.5 e posterior

A propriedade `referencesStatus` reporta a integridade criptográfica das referências no elemento SignedData da assinatura.

- `valid` — Se a compilação computada de cada referência na assinatura corresponder à compilação registrada na assinatura XML. Um status `valid` indica que os dados assinados não foram alterados.
- `invalid` — Se qualquer compilação computada não corresponder à compilação da assinatura.

- `unknown` — Se as compilações de referência não tiverem sido verificadas. As referências não são verificadas se a compilação geral da assinatura for `invalid` ou o certificado de assinatura for inválido. Se `identityStatus` for `unknown`, então as referências serão verificadas somente quando `referencesValidationSetting` for `validOrUnknown`.

Sobre assinaturas XML

Adobe AIR 1.5 e posterior

Uma assinatura XML é uma assinatura digital representada em sintaxe XML. Os dados de uma assinatura XML podem ser usados para validar as informações assinadas que não tenham sido alteradas desde sua realização. Além disso, quando um certificado de assinatura é emitido por uma autoridade de certificação confiável, a identidade de quem assina pode ser verificada por meio de uma infraestrutura de chave pública.

É possível aplicar uma assinatura XML a qualquer tipo de dado digital (em formato binário ou XML). Assinaturas XML geralmente são usadas para finalidades como:

- verificar se recursos externos ou baixados foram modificados
- verificar se mensagens vêm de uma fonte conhecida
- validar privilégios de licença de aplicativo ou assinatura

Sintaxe de assinatura XML com suporte

Adobe AIR 1.5 e posterior

O AIR oferece suporte aos seguintes elementos da recomendação do W3C para processamento e sintaxe de assinatura XML:

- Todos os elementos principais de sintaxe de assinatura (seção 4 do documento de recomendação do W3C), exceto o elemento KeyInfo, que não é totalmente suportado.
- O elemento KeyInfo deve conter somente um elemento X509Data
- Um elemento X509Data deve conter um elemento X509Certificate
- O método de compilação SHA256
- O algoritmo de assinatura RSA-SHA1 (PKCS1)
- O Método e transformação de canonização "XML canônico sem comentários".
- A transformação da assinatura envelopada
- carimbos de data/hora

O documento a seguir ilustra uma assinatura XML típica (a maioria dos dados criptografados não foi removida para simplificar o exemplo):

```
<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>
```

Os elementos chave de uma assinatura são:

- SignedInfo — Contém referências aos dados assinados e os valores de compilação computados no momento da assinatura. Os dados assinados podem ser incluídos no mesmo documento que a assinatura XML ou podem ser externos.
- SignatureValue — Contém uma compilação do elemento SignedInfo criptografado com a chave privada de quem assina.
- KeyInfo — Contém o certificado de assinatura, assim como certificados adicionais necessários para estabelecer a cadeia de confiança. Observe que, embora o elemento KeyInfo seja tecnicamente opcional, o AIR não pode validar a assinatura se ele não estiver incluído.

Há três tipos gerais de assinatura XML:

- Envelopada - a assinatura é inserida nos dados XML que ela está assinando.
- Envelopando - os dados XML assinados estão contidos em um elemento Object do elemento Signature.
- Desanexada - os dados assinados são externos à assinatura XML. Os dados assinados podem estar em um arquivo externo. Como alternativa, podem estar no mesmo documento XML que a assinatura, mas não como um elemento pai ou filho do elemento Signature.

As assinaturas XML usam URIs para referenciar dados assinados. Os aplicativos de assinatura e validação devem usar as mesmas convenções para resolver essas URIs. Ao usar a classe XMLSignatureValidator, é necessário fornecer uma implementação da interface IURIDereferencer. Essa implementação é responsável por resolver a URI e retornar os dados assinados como objeto ByteArray. O objeto ByteArray retornado é compilado usando o mesmo algoritmo que produziu a compilação na assinatura.

Certificados e confiança

Adobe AIR 1.5 e posterior

Um certificado consiste em uma chave pública, identificando informações e, possivelmente, um ou mais certificados que pertencem à autoridade de certificação emissora.

Há duas formas de estabelecer confiança em um certificado. Você pode estabelecer confiança obtendo uma cópia do certificado diretamente de quem assina, por exemplo, na mídia física, ou por meio de uma transmissão digital segura, como uma transação SSL. Você também pode contar com uma autoridade de certificação para determinar se o certificado de assinatura é confiável.

Para isso, o certificado de assinatura deve ser emitido por uma autoridade confiável no computador em que a assinatura é validada. A maioria dos fabricantes de sistemas operacionais insere os certificados raiz de várias autoridades de certificação no armazenamento confiável do sistema operacional. Os usuários também podem adicionar e remover certificados da armazenagem.

Mesmo se um certificado for emitido por uma autoridade de certificação confiável, você deve decidir se ele pertence a alguém em quem confia. Em muitos casos, essa decisão é transmitida ao usuário final. Por exemplo, quando um aplicativo do AIR é instalado, o instalador do AIR exibe as informações de identificação do certificado do editor ao solicitar ao usuário que verifique se deseja instalar o aplicativo. Em outros casos, pode ser necessário comparar a chave pública ou outras informações do certificado com uma lista de chaves aceitáveis. (Essa lista deve ser protegida, talvez por sua própria assinatura ou no armazenamento local criptografado do AIR, de forma que não seja possível adulterá-la).

Nota: *Embora seja possível optar por confiar no certificado de assinatura sem verificação independente, como quando uma assinatura é "autoassinada", você não obtém muita segurança verificando a assinatura. Sem saber quem criou uma assinatura, a garantia de que ela não tenha sido adulterada é, no mínimo, sem valor. A assinatura pode ser uma falsificação assinada validamente.*

Expiração e revogação do certificado

Adobe AIR 1.5 e posterior

Todos os certificados expiram. Eles também podem ser revogados pela autoridade de certificação que o emitiram se, por exemplo, a chave privada relacionada ao certificado estiver comprometida ou tiver sido roubada. Se uma assinatura for executada com um certificado revogado ou expirado, será reportada como inválida, a menos que um carimbo de data e hora tenha sido incluído como parte dela. Se houver um carimbo de data e hora, a classe XMLSignatureValidator validará a assinatura desde que o certificado esteja válido no momento da assinatura.

Um carimbo de data e hora é uma mensagem digital assinada de um serviço de carimbo de data e hora de que os dados foram assinados em uma data e hora específicas. Eles são emitidos por autoridades de carimbo de data e hora e assinados pelo próprio certificado da autoridade do carimbo de data e hora. O certificado da autoridade de carimbo de data e hora incorporada no carimbo de data e hora deve ser confiável no computador atual para que o carimbo de data e hora seja considerado válido. O XMLSignatureValidator não fornece uma API para designar um certificado diferente para usar na validação do carimbo de data e hora.

Implementação da interface IURIDereferencer

Adobe AIR 1.5 e posterior

Para validar uma assinatura XML, você deve fornecer uma implementação da interface IURIDereferencer. A implementação é responsável por resolver as URIs dentro dos elementos Reference de um documento de assinatura XML e retornar os dados de forma que seja possível computar a compilação. A compilação computada é comparada com a compilação da assinatura para determinar se os dados referenciados foram alterados desde a criação da assinatura.

Nota: *Aplicativos do AIT baseados em HTML devem importar uma biblioteca SWF contendo uma implementação ActionScript para validar assinaturas XML. Não é possível implementar a interface IURIDereferencer em JavaScript.*

A interface IURIDerefencer tem um único método, `dereference(uri:String)`, que deve ser implementado. O objeto XMLSignatureValidator chama esse método para cada referência na assinatura. O método deve retornar os dados em um objeto ByteArray.

Na maioria dos casos, você também precisará adicionar propriedades ou métodos que permitam cancelar referência ao objeto para localizar os dados referenciados. Por exemplo, se os dados assinados estiverem localizados no mesmo documento da assinatura, você pode adicionar uma variável membro que forneça uma referência ao documento XML. O método `dereference()` pode usar essa variável, juntamente com a URI, para localizar os dados referenciados. Da mesma forma, se os dados assinados estiverem localizados em um diretório do sistema de arquivos locais, o método `dereference()` poderá precisar de uma propriedade que forneça o caminho para o diretório a fim de resolver os arquivos referenciados.

O XMLSignatureValidator conta inteiramente com o cancelador de referências para interpretar as sequências de caracteres da URI. As regras padrão para cancelar referências de URIs são apresentadas na seção 4.3.3 das Recomendações do W3C para processamento e sintaxe de assinatura XML.

Cancelamento de referência de URIs em assinaturas envelopadas

Adobe AIR 1.5 e posterior

Quando uma assinatura XML envelopada é gerada, os elementos da assinatura são inseridos nos dados assinados. Por exemplo, se você tiver assinado a mensagem a seguir usando uma estrutura de assinatura envelopada:

```
<message>
    <data>...</data>
</message>
```

O documento assinado resultado terá esta aparência:

```
<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>
```

Observe que a assinatura contém um único elemento Reference com uma cadeia de caracteres vazia como sua URI. Uma cadeia de caracteres vazia nesse contexto se refere à raiz do documento.

Note também que o algoritmo de transformação especifica que uma transformação de assinatura envelopada foi aplicada. Quando isso ocorre, o XMLSignatureValidator remove automaticamente a assinatura do documento antes da computação da compilação. Isso significa que o cancelador de referência não precisa remover o elemento Signature quando retorna os dados.

O exemplo a seguir ilustra um cancelador de referência para assinaturas envelopadas:

```
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;
            }
        }
    }
}
```

Essa classe do cancelador de referências usa uma função construtora com um parâmetro, o `signedMessage`, para tornar o documento com assinatura envelopada disponível para o método `dereference()`. Como a referência em uma assinatura envelopada sempre se refere à raiz dos dados assinados, o método `dereferencer()` grava o documento em uma matriz de bytes e o retorna.

Cancelamento de referência de URIs em assinaturas envelopadas e desanexadas

Adobe AIR 1.5 e posterior

Quando os dados assinados são localizados no mesmo documento que a própria assinatura, as URIs das referências geralmente usam a sintaxe XPath ou XPointer para lidar com elementos assinados. As recomendações do W3C para processamento e sintaxe de assinatura XML recomendam somente essa sintaxe, portanto você deve basear sua implementação nas assinaturas que espera encontrar (e adicionar verificação de erro suficiente par lidar bem com a sintaxe sem suporte).

A assinatura de um aplicativo do AIR é um exemplo de uma assinatura envelopada. Os arquivos no aplicativo são listados em um elemento Manifest. O elemento Manifest é tratado no atributo Reference da URI usando a sequência de caracteres “#PackageContents”, que se refere à Id do elemento Manifest:

```
<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>
```

Um cancelador de referência para validar essa assinatura deve usar a sequência de caracteres da URI contendo `"#PackageContents"` do elemento Reference e retornar o elemento Manifest em um objeto ByteArray. O símbolo "#" se refere ao valor de um atributo da Id do elemento.

O exemplo a seguir implementa um cancelador de referência para validar assinaturas do aplicativo do AIR. A implementação é mantida simples contando com a estrutura conhecida de uma assinatura do AIR. Um cancelador de referência de finalidade geral pode ser bem mais complexo.

```
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;
            }

        }
    }
}
```

Quando você verifica esse tipo de assinatura, somente os dados do elemento Manifest são validados. Os arquivos reais do pacote não são verificados. Para verificar se os arquivos do pacote estão adulterados, é necessário ler os arquivos, computar a compilação SHA256 e comparar o resultado à compilação registrada no manifesto. O XMLSignatureValidator não verifica automaticamente essas referência secundárias.

Nota: *Esse exemplo é fornecido somente para ilustrar o processo de validação da assinatura. Há pouco uso em um aplicativo do AIR que valida a própria assinatura. Se o aplicativo já tiver sido adulterado, o agente de adulteração pode simplesmente remover a verificação de validação.*

Computação de valores de compilação para recursos externos

Adobe AIR 1.5 e posterior

O AIR não inclui funções internas para computar compilações SHA256, mas o FlexSDK não inclui uma classe de utilitário SHA256. O SDK inclui também a classe de utilitário do codificador Base64 que é útil para comparar a compilação computada com a compilação armazenada em uma assinatura.

O exemplo de função a seguir lê e valida os arquivos de um manifesto do pacote do AIR:

```
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;
}
```

A função realiza um loop por todas as referências do elemento Manifest. Para cada referência, a compilação SHA256 é computada, codificada no formato base64 e comparada com a compilação no manifesto. As URIs no pacote do AIR se referem aos caminhos relativos ao diretório do aplicativo. Os caminhos são resolvidos com base na localização do arquivo de assinatura, que está sempre no subdiretório META-INF dentro do diretório do aplicativo. Observe que a classe Flex SHA256 retorna a compilação como sequência de caracteres hexadecimais. Essa sequência de caracteres deve ser convertida em um ByteArray contendo os bytes representados pela sequência de caracteres hexadecimais.

Para usar as classes mx.utils.SHA256 e Base64Encoder no Flash CS4, você pode localizar e copiar tais classes no diretório de desenvolvimento do seu aplicativo ou compilar uma biblioteca de SWF contendo as classes que usam o SDK do Flex.

Cancelamento de referência de URIs em assinaturas desanexadas que referenciam dados externos

Adobe AIR 1.5 e posterior

Quando uma URI referencia um recurso externo, os dados devem ser acessados e carregados em um objeto ByteArray. Se a URI contiver uma URL absoluta, será uma simples questão de ler um arquivo ou solicitar uma URL. Se, como ocorre com a maioria dos casos, a URI contiver um caminho relativo, sua implementação IURIDereferencer deverá incluir uma forma de resolver os caminhos para arquivos assinados.

O exemplo a seguir usa um objeto File inicializado quando a instância do cancelador de referência é construída como base para resolver arquivos assinados.

```
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;
            }
        }
    }
}
```

A função `dereference()` simplesmente localiza o arquivo endereçado pela URI de referência, carrega o conteúdo do arquivo em uma matriz de bytes e retorna o objeto ByteArray.

Nota: *Antes de validar referências externas remotas, considere a possibilidade de o aplicativo estar vulnerável a um "phone home" ou um tipo de ataque semelhante por um documento de assinatura construído maliciosamente.*

Capítulo 49: Ambiente do sistema cliente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Essa discussão explica como interagir com o sistema do usuário. Mostra como determinar quais recursos possuem suporte e como criar aplicativos em diversos idiomas utilizando o editor de método de entrada (IME) do usuário, quando disponível. Ele também mostra usuários típicos para domínios de aplicativo.

Mais tópicos da Ajuda

flash.system.System

flash.system.Capabilities

Noções básicas do ambiente do sistema cliente

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Conforme você cria aplicativos mais avançados, talvez ache necessário conhecer detalhes — e funções de acesso — dos sistemas operacionais dos usuários. O pacote flash.system contém uma coleção de classes que permitem acessar funcionalidades de nível de sistema:

- Determinando qual aplicativo e código de domínio de segurança é executado no
- Determinar as capacidades da ocorrência do tempo de execução do Flash (como Flash® Player ou Adobe® AIR™) do usuário, como o tamanho da tela (resolução), e se determinadas funcionalidades estão disponíveis, como áudio mp3
- Criar sites multilíngues usando o IME.
- Interagir com o contêiner do tempo de execução do Flash (que pode ser uma página HTML ou um aplicativo de contêiner).
- Salvar informações na área de transferência do usuário.

O pacote flash.system também inclui as classes IMEConversionMode e SecurityPanel. Essas classes contêm constantes estáticas que são usadas com as classes IME e Security, respectivamente.

Conceitos e termos importantes

A lista de referências a seguir contém termos importantes:

Sistema operacional O programa principal executado em um computador, no qual todos os outros aplicativos são executados, como o Microsoft Windows, Mac OS X ou Linux®.

Clipboard O contêiner do sistema operacional onde manter texto ou itens que são copiados ou recortados e do qual os itens são colados nos aplicativos.

Domínio do aplicativo Um mecanismo para separação de classes usadas em diferentes arquivos SWF, de forma que os arquivos SWF incluam diferentes classes com o mesmo nome, as classes não são substituídas umas pelas outras.

IME (editor do método de entrada) Um programa (ou ferramenta do sistema operacional) usado para inserir caracteres ou símbolos complexos usando um teclado padrão.

Sistema cliente Em termos de programação, um cliente é a parte de um aplicativo (ou todo o aplicativo) que é executado no computador individual e é usada por um único usuário. O sistema cliente é o sistema operacional subjacente no computador do usuário.

Uso da classe System

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe System contém métodos e propriedades que permitem interagir com o sistema operacional do usuário e recuperar a utilização de memória atual do tempo de execução. Os métodos e as propriedades da classe System também permitem ouvir eventos `imeComposition`, instruir o tempo de execução do a carregar arquivos de texto externos usando a página de código atual do usuário ou carregá-las como Unicode ou definir o conteúdo da área de transferência do usuário.

Obtenção de dados sobre o sistema do usuário em tempo de execução

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Com a verificação da propriedade `System.totalMemory`, é possível determinar a quantidade de memória (em bytes) que o tempo de execução do está usando no momento. Essa propriedade permite monitorar o uso de memória do monitor e otimizar os aplicativos com base em como o nível de memória é alterado. Por exemplo, se um efeito visual específico provocar um grande aumento no uso de memória, você poderá desejar considerar modificar ou eliminar o efeito completamente.

A propriedade `System.ime` é uma referência ao IME (Editor de método de entrada) instalado. Essa propriedade permite ouvir eventos `imeComposition` (`flash.events.IMEEvent.IME_COMPOSITION`) usando o método `addEventListener()`.

A terceira propriedade na classe System é `useCodePage`. Quando `useCodePage` está definido como `true`, o tempo de execução do usa a página de código tradicional do sistema operacional para carregar os arquivos de texto externos. Se definir essa propriedade como `false`, você indicará ao tempo de execução do para interpretar o arquivo externo como Unicode.

Se você definir `System.useCodePage` como `true`, lembre-se de que a página de código tradicional do sistema operacional deve incluir os caracteres usados no arquivo de texto externo para que o texto seja exibido. Por exemplo, se você carregar um arquivo de texto externo com caracteres chineses, esses caracteres não poderão ser exibidos em um sistema que use a página de código do Windows em inglês, pois essa página de código não inclui caracteres chineses.

Para garantir que usuários em diversas plataformas visualizem arquivos de texto externos utilizados em seu aplicativo, codifique todos os arquivos texto como Unicode e mantenha `System.useCodePage` definido em`false` com padrão. Dessa forma, o tempo de execução do interpreta o texto como Unicode.

Salvamento de texto na área de transferência

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe System inclui um método chamado `setClipboard()` que permite que o tempo de execução do Flash defina o conteúdo da área de transferência do usuário com uma string especificada. Por motivos de segurança, não existe nenhum método `Security.getClipboard()`, pois esse método pode potencialmente permitir que sites mal-intencionados acessem os dados copiados recentemente na área de transferência do usuário.

O código a seguir ilustra como uma mensagem de erro pode ser copiada na área de transferência do usuário quando ocorre um erro de segurança. A mensagem de erro poderá ser útil se o usuário desejar relatar um bug potencial com um aplicativo.

```
private function securityErrorHandler(event:SecurityErrorEvent):void
{
    var errorString:String = "[" + event.type + "] " + event.text;
    trace(errorString);
    System.setClipboard(errorString);
}
```

Flash Player 10 e AIR 1.0

Você pode usar a classe Clipboard para ler e gravar dados na área de transferência e reagir a um evento do usuário. No AIR, um evento de usuário não é obrigatório para código executado na área de segurança do aplicativo para acessar a área de transferência.

Uso da classe Capabilities

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe Capabilities permite que os desenvolvedores determinem o ambiente no qual o aplicativo está sendo executado. Com o uso de várias propriedades da classe Capabilities, é possível descobrir a resolução do sistema do usuário, se o sistema do usuário oferece suporte a software de acessibilidade, o idioma do sistema operacional do usuário, bem como a versão do tempo de execução do Flash.

Verificando as propriedade da classe Capabilities, é possível personalizar o aplicativo para trabalhar melhor com o ambiente específico do usuário. Por exemplo, verificando as propriedades `Capabilities.screenResolutionX` e `Capabilities.screenResolutionY`, é possível determinar a resolução de vídeo que o sistema do usuário está usando e decidir qual tamanho de vídeo pode ser mais apropriado. Ou é possível verificar a propriedade `Capabilities.hasMP3` para verificar se o sistema do usuário oferece suporte à reprodução de mp3 antes de tentar carregar um arquivo mp3 externo.

O código a seguir usa uma expressão regular para analisar a versão do tempo de execução do Flash que está sendo usada pelo cliente:

```
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.");
}
```

Para enviar as capacidades do sistema do usuário para um script do lado do servidor de forma que as informações sejam armazenadas em um banco de dados, é possível usar o seguinte código ActionScript:

```
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);
```

Exemplo de recursos: detecção de capacidades do sistema

Flash Player 9 e posterior

O exemplo CapabilitiesExplorer demonstra como usar a classe flash.system.Capabilities para determinar quais recursos são suportados pela versão do usuário do tempo de execução do Flash. Este exemplo ensina as seguintes técnicas:

- Detecção das capacidades suportadas pela versão do usuário do tempo de execução do Flash usando a classe Capabilities.
- Uso da classe ExternalInterface para detectar quais configurações de navegador são suportadas pelo navegador do usuário.

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo CapabilitiesExplorer podem ser encontrados na pasta Amostras/CapabilitiesExplorer. Esse aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
CapabilitiesExplorer.fla ou CapabilitiesExplorer.mxml	O arquivo principal do aplicativo no Flash (FLA) ou no Flex (MXML).
com/example/programmingas3/capabilities/CapabilitiesGrabber.as	A classe que fornece a funcionalidade principal do aplicativo, incluindo a adição das capacidades do sistema a uma matriz, a classificação dos itens e o uso da classe ExternalInterface para recuperar as capacidades do navegador.
capabilities.html	Um contêiner HTML que contém o JavaScript necessário para se comunicar com a API externa.

Visão geral de CapabilitiesExplorer

Flash Player 9 e posterior

O arquivo CapabilitiesExplorer.mxml é responsável por configurar a interface do usuário para o aplicativo CapabilitiesExplorer. As capacidades da versão do usuário do tempo de execução do Flash serão exibidas dentro de uma ocorrência do componente DataGrid no Palco. As capacidades do navegador também serão exibidas se eles estiverem executando o aplicativo em um contêiner HTML e se a API externa estiver disponível.

Quando o evento `creationComplete` do arquivo do aplicativo principal é despachado, o método `initApp()` é chamado. O método `initApp()` chama o método `getCapabilities()` de dentro da classe com.example.programmingas3.capabilities.CapabilitiesGrabber. O código do método `initApp()` é o seguinte:

```
private function initApp():void
{
    var dp:Array = CapabilitiesGrabber.getCapabilities();
    capabilitiesGrid.dataProvider = dp;
}
```

O método `CapabilitiesGrabber.getCapabilities()` retorna uma matriz classificada do tempo de execução do Flash e as capacidades do navegador que, em seguida, são definidas para a propriedade `dataProvider` da ocorrência do componente DataGrid `capabilitiesGrid` no Palco.

Visão geral da classe CapabilitiesGrabber

Flash Player 9 e posterior

O método estático `getCapabilities()` da classe CapabilitiesGrabber adiciona cada propriedade da classe flash.system.Capabilities a uma matriz (`capDP`). Em seguida, ele chama o método estático `getBrowserObjects()` na classe CapabilitiesGrabber. O método `getBrowserObjects()` usa a API externa para loop sobre o objeto navigator do navegador que contém as capacidades do navegador. O método `getCapabilities()` é o seguinte:

```
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;
}
```

O método `getBrowserObjects()` retorna uma matriz de cada uma das propriedades do objeto navigator no navegador. Se essa matriz tiver um comprimento de um ou mais itens, a matriz de capacidades do navegador (`navArr`) será anexada à matriz de capacidades do Flash Player (`capDP`) e a matriz inteira será classificada alfabeticamente. Finalmente, a matriz classificada será retornada ao arquivo do aplicativo principal que, em seguida preencherá a grade de dados. O código para o método `getBrowserObjects()` é o seguinte:

```
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;
}
```

Se a API externa estiver disponível no ambiente atual do usuário, o tempo de execução do Flash chamará o método `JS_getBrowserObjects()` do JavaScript que executa loop sobre o objeto navigator do navegador e retorna uma string de valores codificados de URL ao ActionScript. Em seguida, essa string é convertida em um objeto URLVariables (`itemVars`) e adicionada à matriz `itemArr` que é retornada para o script de chamada.

Comunicação com o JavaScript

Flash Player 9 e posterior

A parte final da criação do aplicativo CapabilitiesExplorer é a escrita do JavaScript necessário para executar loop sobre cada um dos itens no objeto navigator do navegador e anexar um par de nome e valor a uma matriz temporária. O código do método `JS_getBrowserObjects()` do JavaScript no container.html é o seguinte:

```
<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>
```

O código começa com a criação de uma matriz temporária que conterá todos os pares de nome e valor no objeto navigator. Em seguida, um loop é executado no objeto navigator usando um loop`for..in` e o tipo de dados do valor atual é avaliado para filtrar valores indesejados. Neste aplicativo, estamos interessados somente nos valores de string ou boolianos e outros tipos de dados (como funções ou matrizes) são ignorados. Cada valor de string ou booliano no objeto navigator é anexado à matriz `tempArr`. Em seguida, é executado um loop no objeto da tela do navegador usando um `for..in` e cada valor numérico é adicionado à matriz `tempArr`. Finalmente, a matriz temporária é convertida em uma string usando o método `Array.join()`. A matriz usa um e comercial (&) como um delimitador, o que permite que o ActionScript analise os dados facilmente usando a classe URLVariables.

Capítulo 50: Chamada e encerramento do aplicativo do AIR

Adobe AIR 1.0 e posterior

Essa seção discute as formas através das quais um aplicativo Adobe® AIR® instalado pode ser invocado, bem como opções e considerações para fechar um aplicativo em execução.

Nota: *Os objetos NativeApplication, InvokeEvent e BrowserInvokeEvent só estão disponíveis para o conteúdo SWF em execução na caixa de proteção do aplicativo do AIR. O conteúdo SWF que está sendo reproduzido no tempo de execução do Flash Play, dentro do navegador ou no player autônomo (projetor), ou em um aplicativo do AIR fora da caixa de proteção do aplicativo, não pode acessar essas classes.*

Para ver uma rápida explicação e exemplos de código da chamada e encerramento de aplicativos AIR, consulte os seguintes artigos de início rápido no Adobe Developer Connection:

- Opções de inicialização

Mais tópicos da Ajuda

flash.desktop.NativeApplication

flash.events.InvokeEvent

flash.events.BrowserInvokeEvent

Chamada do aplicativo

Adobe AIR 1.0 e posterior

Um aplicativo do AIR é invocado quando o usuário (ou o sistema operacional):

- Inicia o aplicativo do shell da área de trabalho.
- Usa o aplicativo como um comando em um shell de linha de comando.
- Abre um tipo de arquivo para o qual o aplicativo é o aplicativo de abertura padrão.
- (Mac OS X) clica no ícone do aplicativo na barra de tarefas do encaixe (esteja o aplicativo em execução ou não no momento).
- Opta por inicializar o aplicativo do instalador (na extremidade de um novo processo de instalação ou após clicar duas vezes no arquivo do AIR para um aplicativo já instalado).
- Começa uma atualização de um aplicativo do AIR quando a versão instalada tiver indicado que está lidando sozinha com atualizações do aplicativo (incluindo uma declaração `<customUpdateUI>true</customUpdateUI>` no arquivo de descrição do aplicativo).
- (iOS) Recebe uma notificação do serviço de Notificação por Push da Apple (APNs).
- Invoca o aplicativo por um URL.

- Visita uma página da Web que hospeda um crachá ou aplicativo do Flash que chama o método `com.adobe.air.AIR launchApplication()` especificando as informações de identificação para o aplicativo do AIR. (O descritor do aplicativo também deve incluir uma declaração `<allowBrowserInvocation>true</allowBrowserInvocation>` para que a chamada do navegador seja bem-sucedida.)

Sempre que um aplicativo do AIR for invocado, o AIR despacha um objeto de tipo InvokeEvent `invoke` pelo objeto singleton NativeApplication. Para permitir que o tempo de um aplicativo se inicialize e registre um ouvinte de evento, eventos `invoke` são enfileirados em vez de descartados. Assim que um ouvinte é registrado, todos os eventos enfileirados são entregues.

Nota: *Quando um aplicativo é chamado usando o recurso de chamada do navegador, o objeto NativeApplication emite somente um evento `invoke` se o aplicativo ainda não estiver em execução.*

Para receber eventos `invoke`, chame o método `addEventListener()` do objeto NativeApplication (`NativeApplication.nativeApplication)`. Quando um ouvinte de evento registra um evento `invoke`, ele também recebe todos os eventos `invoke` que ocorreram antes do registro. Eventos `invoke` enfileirados são despachados um de cada vez em um curto intervalo após a chamada para `addEventListener()` ser retornada. Se um novo evento `invoke` ocorrer durante esse processo, ele poderá ser despachado antes de um ou mais dos eventos enfileirados. Esse enfileiramento de eventos permite que você manipule qualquer evento `invoke` que tenha ocorrido antes de seu código de inicialização ser executado. Tenha em mente que, se você adicionar um ouvinte de evento depois na execução (depois da inicialização do aplicativo), ele ainda receberá todos os eventos `invoke` que ocorreram desde que o aplicativo foi iniciado.

Apenas uma instância de um aplicativo do AIR é iniciada. Quando um aplicativo já em execução é invocado novamente, o AIR despacha um novo evento `invoke` para a instância em execução. É de responsabilidade de um aplicativo do AIR responder a um evento `invoke` e executar a ação apropriada (como abrir uma janela de um novo documento).

Um objeto `InvokeEvent` contém qualquer argumento transmitido ao aplicativo, bem como o diretório a partir do qual o aplicativo foi invocado. Se o aplicativo foi invocado devido a uma associação de tipo de arquivo, todo o caminho para o arquivo será incluído nos argumentos de linha de comando. Da mesma forma, se o aplicativo foi invocado devido a uma atualização de aplicativo, todo o caminho para o arquivo do AIR atualizado será fornecido.

Quando vários arquivos são abertos em uma operação, um único objeto InvokeEvent é despachado no Mac OS X. Cada arquivo é incluído na matriz `arguments`. No Windows e no Linux, um objeto InvokeEvent distinto é despachado para cada arquivo.

Seu aplicativo pode manipular eventos `invoke` registrando um ouvinte com seu objeto NativeApplication:

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvokeEvent);
```

E definindo um ouvinte de evento:

```
 var arguments:Array;
var currentDir:File;
public function onInvokeEvent(invocation:InvokeEvent):void {
    arguments = invocation.arguments;
    currentDir = invocation.currentDirectory;
}
```

Captura de argumentos de linha de comando

Adobe AIR 1.0 e posterior

Os argumentos de linha de comando associados à invocação de um aplicativo do AIR são entregues no objeto InvokeEvent despachado pelo objeto NativeApplication. A propriedade `arguments` do InvokeEvent contém uma matriz dos argumentos transmitidos pelo sistema operacional quando um aplicativo do AIR é invocado. Se os argumentos contêm caminhos de arquivos relativos, você pode normalmente resolver os caminhos usando a propriedade `currentDirectory`.

Os argumentos transmitidos a um programa do AIR são tratados como sequências delimitadas de espaço em branco, a menos que estejam entre aspas duplas:

Argumentos	Matriz
tick tock	{tick,tock}
tick "tick tock"	{tick,tick tock}
"tick" “tock”	{tick,tock}
\"tick\" \"tock\"	{"tick","tock"}

A propriedade `currentDirectory` de um objeto InvokeEvent contém um objeto File que representa o diretório a partir do qual o aplicativo foi iniciado.

Quando um aplicativo é invocado porque um arquivo de um tipo registrado pelo aplicativo é aberto, o caminho nativo para o arquivo é incluído nos argumentos de linha de comando como uma sequência de caracteres. (Seu aplicativo é responsável por abrir ou executar a operação pretendida no arquivo.) Da mesma forma, quando um aplicativo é programado para se atualizar (em vez de confiar na interface de usuário de atualização do AIR padrão), o caminho nativo para o arquivo do AIR é incluído quando um usuário clica duas vezes em um arquivo do AIR que contém um aplicativo com uma ID correspondente do aplicativo.

Você pode acessar o arquivo usando o método `resolve()` do objeto File `currentDirectory`:

```
 if((invokeEvent.currentDirectory != null)&&(invokeEvent.arguments.length > 0)){
    dir = invokeEvent.currentDirectory;
    fileToOpen = dir.resolvePath(invokeEvent.arguments[0]);
}
```

Você também deve validar se um argumento é realmente um caminho para um arquivo.

Exemplo: log de eventos de invocação

Adobe AIR 1.0 e posterior

O exemplo a seguir demonstra como registrar ouvintes para e manipular o evento `invoke`. O exemplo registra todos os eventos de invocação recebidos e exibe o diretório atual e os argumentos de linha de comando.

Exemplo do ActionScript

```
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);
        }
    }
}
```

Exemplo do Flex

```
<?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>
```

Invocando um aplicativo do AIR no logon do usuário

Adobe AIR 1.0 e posterior

Um aplicativo do AIR pode ser definido para ser inicializado automaticamente quando o usuário atual faz login definindo a propriedade `StartAtLogi` do NativeApplication como `true`. Depois de definido, o aplicativo é iniciado automaticamente sempre que o usuário fizer login. Ele continua a ser aberto no logon até que a configuração seja alterada para `false`, o usuário altera manualmente a configuração pelo sistema operacional ou o aplicativo é desinstalado. A inicialização no logon é uma configuração de tempo de execução. A configuração se aplica apenas ao usuário atual. O aplicativo deve ser instalado para definir com êxito a propriedade `startAtLogin` como `true`. Um erro é lançado se a propriedade for definida quando um aplicativo não está instalado (como quando ele é inicializado com o ADL).

Nota: *O aplicativo não é iniciado quando o sistema do computador é iniciado. Ele é iniciado quando o usuário faz login.*

Para determinar se um aplicativo foi iniciado automaticamente ou como resultado de uma ação do usuário, você pode examinar a propriedade `reason` do objeto InvokeEvent. Se a propriedade foi igual a `InvokeEventReason.LOGIN`, então o aplicativo é iniciado automaticamente. Para outros caminhos de invocação, a propriedade `reason` é definida da seguinte maneira:

- `InvokeEventReason.NOTIFICATION` (somente iOS) - O aplicativo foi invocado por APNs. Para obter mais informações sobre APNs, consulte Usar notificações por push.
- `InvokeEventReason.OPEN_URL` - O aplicativo foi invocado por outro aplicativo ou pelo sistema.
- `InvokeEventReason.Standard` - Todos os demais casos.

Para acessar a propriedade `reason`, seu aplicativo deve se focalizar no AIR 1.5.1 ou superior (definindo o valor correto de namespace no arquivo de descrição do aplicativo).

O aplicativo simplificado a seguir usa a propriedade reason do InvokeEvent para decidir como se comportar quando ocorre um evento invoke. Se a propriedade reason for "login", o aplicativo continua em plano de fundo. Do contrário, ele torna o aplicativo principal visível. O aplicativo que usa esse padrão em geral é iniciado no logon, de forma que possa realizar o processamento de fundo ou o monitoramento do evento e abre uma janela em resposta a um evento invoke disparador pelo usuário.

```
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();
            }
        }
    }
}
```

Nota: *Para ver a diferença no comportamento, empacote e instale o aplicativo. A propriedade* `startAtLogin` *só pode ser definida para os aplicativos instalados.*

Invocação de um aplicativo do AIR do navegador

Adobe AIR 1.0 e posterior

Usando o recurso de invocação do navegador, um site da Web pode inicializar um aplicativo instalado do AIR do navegador. A chamada do navegador é permitida apenas se o arquivo descritor do aplicativo definir `allowBrowserInvocation` como `true`:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Quando o aplicativo é invocado pelo navegador, o objeto NativeApplication do aplicativo despacha um objeto BrowserInvokeEvent.

Para receber eventos BrowserInvokeEvent, chame o método `addEventListener()` do objeto NativeApplication (`NativeApplication.nativeApplication`) no aplicativo do AIR. Quando um ouvinte de evento registra um evento BrowserInvokeEvent, ele também recebe todos os eventos BrowserInvokeEvent que ocorreram antes do registro. Esses eventos são despachados depois que a chamada a `addEventListener()` retorna, mas não necessariamente antes de outros eventos BrowserInvokeEvent, que podem ser recebidos depois do registro. Isso permite que você manipule os eventos BrowserInvokeEvent ocorridos antes que seu código de inicialização fosse executado (por exemplo, quando o aplicativo foi inicialmente invocado do navegador). Tenha em mente que, se você adicionar um ouvinte de evento depois na execução (depois da inicialização do aplicativo), ele ainda receberá todos os eventos BrowserInvokeEvent ocorridos desde que o aplicativo foi iniciado.

O objeto BrowserInvokeEvent inclui as seguintes propriedades:

Propriedade	**Descrição**
argumentos	Uma matriz de argumentos (sequências de caracteres) para transmitir ao aplicativo.
isHTTPS	Se o conteúdo do navegador usa o esquema de URL https (`true`) ou não (`false`).
isUserEvent	Se a invocação do navegador resultou em um evento de usuário (como clique do mouse). No AIR 1.0, este ajuste é sempre `true`; o AIR exige um evento de usuário para o recurso de invocação do navegador.
sandboxType	O tipo de caixa de proteção para o conteúdo do navegador. Valores válidos são definidos como aqueles que podem ser usados na propriedade `Security.sandboxType` e podem ser um dos seguintes: • `Security.APPLICATION` – O conteúdo está na caixa de proteção de segurança do aplicativo. • `Security.LOCAL_TRUSTED` – O conteúdo está na caixa de proteção de segurança local com sistema de arquivos. • `Security.LOCAL_WITH_FILE` – O conteúdo está na caixa de proteção de segurança local com sistema de arquivos. • `Security.LOCAL_WITH_NETWORK` – O conteúdo está na caixa de proteção de segurança local com rede. • `Security.REMOTE` — O conteúdo está em um domínio remoto (de rede).
securityDomain	O domínio de segurança para o conteúdo do navegador, como `"www.adobe.com"` ou `"www.example.org"`. Essa propriedade é definida apenas para conteúdo na caixa de proteção de segurança remota (para conteúdo de um domínio de rede). Ele não é definido para conteúdo em uma caixa de proteção de segurança local ou do aplicativo.

Se você usa o recurso de invocação do navegador, certifique-se de considerar implicações de segurança. Quando um site da Web inicia um aplicativo do AIR, ele pode enviar dados pela propriedade `arguments` do objeto BrowserInvokeEvent. Cuidado ao usar esses dados em qualquer operação confidencial, como carregar código ou arquivo de APIs. Esse nível de risco depende do que o aplicativo está fazendo com os dados. Se você espera apenas um site da Web específico para invocar o aplicativo, o aplicativo deve verificar a propriedade `securityDomain` do objeto BrowserInvokeEvent. Você também pode exigir que o site invoque o aplicativo para usar HTTPs, o que você pode verificar marcando a propriedade `isHTTPS` do objeto BrowserInvokeEvent.

O aplicativo deve validar os dados transmitidos. Por exemplo, se um aplicativo espera para transmitir URLs a um domínio específico, ele deve validar se as URLs realmente apontam para aquele domínio. Isso pode impedir que um invasor engane o aplicativo para lhe enviar dados confidenciais.

Nenhum aplicativo deve usar argumentos BrowserInvokeEvent que possam apontar para recursos locais. Por exemplo, um aplicativo não deve criar objetos File com base em um caminho transmitido do navegador. Se espera-se que caminhos remotos sejam transmitidos do navegador, o aplicativo deve garantir que os caminhos não usem o protocolo `file://` em vez de um protocolo remoto.

Encerramento do aplicativo

Adobe AIR 1.0 e posterior

A forma mais rápida de encerrar um aplicativo é chamar o método exit() do NativeApplication. Isso funciona bem quando seu aplicativo não tem dados para salvar ou recursos externos para limpar. Chamar `exit()` fecha todas as janelas e, em seguida, encerra o aplicativo. No entanto, para permitir que janelas ou outros componentes do seu aplicativo interrompam o processo de encerramento, talvez para salvar dados vitais, despache os eventos de aviso adequados antes de chamar `exit()`.

Outra consideração sobre como fechar um aplicativo graciosamente é fornecer um único caminho de execução, sem importar como o processo de encerramento é iniciado. O usuário (ou sistema operacional) pode disparar o encerramento do aplicativo das seguintes maneiras:

- Fechando a última janela do aplicativo quando `NativeApplication.nativeApplication.autoExit` for `true`.
- Selecionando o comando de saída do aplicativo do sistema operacional; por exemplo, quando o usuário escolhe o comando de sair do aplicativo do menu padrão. (Isso só ocorre com o Mac OS. O Windows e o Linux não fornecem um comando de saída do aplicativo por meio do cromo do sistema.)
- Desligando o computador.

Quando um comando de saída é mediado pelo sistema operacional por uma dessas rotas, o NativeApplication despacha um evento `exiting`. Se nenhum ouvinte cancelar o evento `exiting`, qualquer janela aberta será fechada. Cada janela despacha um evento `closing` e, em seguida, um `close`. Se alguma das janelas cancelar o evento `closing`, o processo de encerramento será interrompido.

Se a ordem do fechamento das janelas for um problema para o seu aplicativo, ouça o evento `exiting` do NativeApplication e feche você mesmo as janelas na ordem adequada. Talvez você precise fazer isso, por exemplo, se tiver uma janela de documento com as paletas da ferramenta. Poderia ser inconveniente, ou pior, se o sistema fechou as paletas, mas o usuário decidiu cancelar o comando de sair para salvar alguns dados. No Windows, o único momento que você obterá o evento `exiting` é depois de fechar a última janela (quando a propriedade `autoExit` do objeto NativeApplication for definida como `true`).

Para fornecer um comportamento consistente em todas as plataformas, seja a sequência de saída iniciada pelo cromo do sistema operacional, por comandos de menu ou pela lógica do aplicativo, observe as seguintes práticas recomendadas para sair do aplicativo:

1. Sempre despache um evento `exiting` pelo objeto NativeApplication antes de chamar `exit()` no código do aplicativo e verifique se outro componente do seu aplicativo não cancela o evento.

```
public function applicationExit():void {
    var exitingEvent:Event = new Event(Event.EXITING, false, true);
    NativeApplication.nativeApplication.dispatchEvent(exitingEvent);
    if (!exitingEvent.isDefaultPrevented()) {
        NativeApplication.nativeApplication.exit();
    }
}
```

2. Ouça o evento `exiting` do aplicativo do objeto `NativeApplication.nativeApplication` e, no manipulador, feche qualquer janela (despachando um evento `closing` primeiro). Execute qualquer tarefa de limpeza necessária, como salvar dados de aplicativo ou excluir arquivos temporários, após todas as janelas terem sido fechadas. Apenas use métodos síncronos durante a limpeza para garantir que eles sejam concluídos antes que o aplicativo seja encerrado.

Se a ordem na qual suas janelas forem fechadas não importar, você pode efetuar um loop pela matriz `NativeApplication.nativeApplication.openedWindows` e fechar cada janela sucessivamente. Se a ordem *importar*, forneça um modo de fechar as janelas na sequência correta.

```
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 As janelas devem sempre manipular sua própria limpeza ouvindo seus próprios eventos `closing`.

4 Use apenas um ouvinte `exiting` no seu aplicativo uma vez que manipuladores chamados anteriormente não podem saber se os manipuladores subsequentes irão cancelar o evento `exiting` (e não seria inteligente confiar na ordem de execução).

Capítulo 51: Trabalho com informações de tempo de execução do AIR e de sistema operacional

Adobe AIR 1.0 e posterior

Essa seção discute formas através das quais um aplicativo do AIR pode gerenciar as associações do arquivo do sistema operacional, detectar a atividade do usuário e obter informações sobre o tempo de execução do Adobe® AIR®.

Mais tópicos da Ajuda

flash.desktop.NativeApplication

Gerenciamento de associações de arquivos

Adobe AIR 1.0 e posterior

Associações entre seu aplicativo e um tipo de arquivo devem ser declaradas no descritor do aplicativo. Durante o processo de instalação, o instalador do aplicativo do AIR associa o aplicativo do AIR como o aplicativo de abertura padrão para cada um dos tipos de arquivo declarados, a menos que outro aplicativo já seja o padrão. O processo de instalação do aplicativo do AIR não substitui uma associação de tipo de arquivo existente. Para se apoderar da associação de outro aplicativo, chame o método `NativeApplication.setAsDefaultApplication()` no tempo de execução.

Constitui boa prática verificar se as associações de arquivos esperadas estão em vigor quando seu aplicativo inicializar Isso porque o instalador do aplicativo do AIR não sobrescreve associações de arquivo existentes e porque associações de arquivo em um sistema de usuário podem mudar a qualquer momento. Quando outro aplicativo tem a associação de arquivo atual, também é boa prática solicitar ao usuário antes de se apoderar de uma associação existente.

Os métodos seguintes da classe NativeApplication permitem que um aplicativo gerencie associações de arquivos. Cada um dos métodos toma a extensão de tipo de arquivo como um parâmetro:

Método	Descrição
isSetAsDefaultApplication()	Retorna true se o aplicativo do AIR está associado atualmente ao tipo de arquivo especificado.
setAsDefaultApplication()	Cria a associação entre o aplicativo do AIR e a ação aberta do tipo de arquivo.
removeAsDefaultApplication()	Remove a associação entre o aplicativo do AIR e o tipo de arquivo.
getDefaultApplication()	Reporta o caminho do aplicativo que está associado atualmente ao tipo de arquivo.

O AIR só pode gerenciar associações para os tipos de arquivo originalmente declarados no descritor do aplicativo. Você não pode obter informações sobre as associações de um tipo de arquivo não declarado, mesmo que um usuário tenha criado manualmente a associação entre esse tipo de arquivo e seu aplicativo. Chamar algum dos métodos de gerenciamento de associação de arquivo com a extensão de um tipo de arquivo não declarado no descritor do aplicativo faz com que o aplicativo lance uma exceção de tempo de execução.

Obtenção da versão do tempo de execução e do nível de patch

Adobe AIR 1.0 e posterior

O objeto NativeApplication tem uma propriedade `runtimeVersion`, que é a versão do tempo de execução em que o aplicativo está sendo executado (uma string, como `"1.0.5"`). O objeto NativeApplication também tem uma propriedade `runtimePatchLevel`, que é o nível de patch do tempo de execução (um número, como 2960). O código seguinte usa estas propriedades:

```
trace(NativeApplication.nativeApplication.runtimeVersion);
trace(NativeApplication.nativeApplication.runtimePatchLevel);
```

Detecção de recursos do AIR

Adobe AIR 1.0 e posterior

Para um arquivo que esteja reunido com o aplicativo Adobe AIR, a propriedade `Security.sandboxType` está definida como o valor definido pela constante `Security.APPLICATION`. Você pode carregar o conteúdo (que pode ou não conter APIs específicas ao AIR) baseado em um arquivo estar na caixa de proteção de segurança do Adobe AIR, como ilustrado no código seguinte:

```
if (Security.sandboxType == Security.APPLICATION)
{
    // Load SWF that contains AIR APIs
}
else
{
    // Load SWF that does not contain AIR APIs
}
```

Todos os recursos não instalado com o aplicativo do AIR são atribuídos às mesmas caixas de proteção de segurança como seriam atribuídos pelo Adobe® Flash® em um navegador da Web. Recursos remotos são colocados em caixas de proteção, de acordo com seus domínios de origem, e recursos locais são colocados na caixa de proteção local com rede, local com sistema de arquivos ou local confiável.

Você pode verificar se a propriedade estática `Capabilities.playerType` está definida como `"Desktop"` para ver se o conteúdo está sendo executado no tempo de execução (e não em execução no Flash Player em execução em um navegador).

Para obter mais informações, consulte “Segurança do AIR” na página 1116.

Rastreamento de presença do usuário

Adobe AIR 1.0 e posterior

O objeto NativeApplication despacha dois eventos que o ajudam a detectar quando o usuário está usando o computador ativamente. Se não for detectada nenhuma atividade do mouse nem do teclado no intervalo de tempo determinado pela propriedade `NativeApplication.idleThreshold`, o NativeApplication despacha um evento `userIdle`. Quando ocorrer a próxima entrada de mouse ou teclado, o objeto NativeApplication despachará um evento `userPresent`. O intervalo `idleThreshold` é medido em segundos e tem um valor padrão de 300 (5 minutos). Você também ode obter o número de segundos desde a última entrada de usuário da propriedade `NativeApplication.nativeApplication.lastUserInput`.

As linhas seguintes de código definem o limite de ociosidade para 2 minutos e ouvem tanto eventos `userIdle` como `userPresent`:

```
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");
});
```

Nota: *Apenas um único evento `userIdle` é despachado entre quaisquer dois eventos `userPresent`.*

Capítulo 52: Trabalho com janelas nativas do AIR

Adobe AIR 1.0 e posterior

Use as classes fornecidas pela API de janela nativa do Adobe® AIR® para criar e gerenciar janelas da área de trabalho.

Princípios básicos das janelas nativas no AIR

Adobe AIR 1.0 e posterior

Para ver uma explicação rápida e exemplos de código para trabalhar com janelas originais no AIR, consulte os seguintes artigos de início rápido do Adobe Developer Connection:

- Criação de um aplicativo com janela transparente (Flex)
- Interação com uma janela (Flex)
- Personalização da aparência de uma janela nativa (Flex)
- Abertura de janelas (Flex)
- Criação de janelas no estilo pop-up (Flex)
- Controle da ordem de exibição das janelas (Flex)
- Criação de janelas não retangulares redimensionáveis (Flex)
- Interação com uma janela (Flex)
- Personalização da aparência de uma janela nativa (Flex)
- Criação de janelas no estilo pop-up (Flash)
- Controle da ordem de exibição das janelas (Flash)
- Criação de janelas não retangulares redimensionáveis (Flash)

O AIR oferece uma API de janelas fácil de usar e compatível com várias plataformas, que permite criar janelas de sistema operacional nativas usando técnicas de programação em Flash®, Flex™ e HTML.

Com o AIR, você tem grande liberdade para desenvolver a aparência do seu aplicativo. As janelas que você cria parecem um aplicativo padrão de área de trabalho, correspondendo ao estilo Apple quando em execução no Mac, em conformidade com convenções Microsoft quando executadas no Windows e em harmonia com o gerenciador de janelas do Linux, tudo sem incluir um código específico de linha de plataforma. Se preferir, você pode usar o cromo extensível e que pode ter a capa trocada, fornecido pela estrutura do Flex, para estabelecer o seu próprio estilo, independentemente do local onde o aplicativo é executado. Você pode até mesmo desenhar seus próprios cromos de janelas usando arte final vetorial e de bitmap com suporte completo para transparência e mesclagem alfa na área de trabalho. Cansado de janelas retangulares? Desenhe uma redonda.

Janelas no AIR

Adobe AIR 1.0 e posterior

O AIR suporta três APIs diferentes para se trabalhar com janelas:

- A classe NativeWindow orientada para ActionScript fornece o nível mais baixo de API de janela. Use o NativeWindows em aplicativos de autoria Flash Professional e ActionScript. Considere estender a classe NativeWindow para especializar as janelas usadas em seu aplicativo.
- No ambiente HTML, você pode usar a classe JavaScript Window, assim como faria em um aplicativo da Web baseado em navegador. As chamadas para os métodos JavaScript Window são encaminhadas para o objeto janela nativo subjacente.
- As classes mx:WindowedApplication e mx:Window da estrutura do Flex fornecem um "invólucro" para a classe NativeWindow. O componente WindowedApplication substitui o componente Application quando você cria um aplicativo do AIR com Flex e deve ser sempre usado como janela inicial no aplicativo Flex.

Janelas ActionScript

Quando criar janelas com a classe NativeWindow, use a lista de exibição no palco do Flash Player diretamente. Para adicionar um objeto visual a uma NativeWindow, adicione o objeto à lista de exibição do palco da janela ou a outro contêiner do objeto de exibição no palco.

Janelas HTML

Quando cria janelas HTML, você usa HTML, CSS e JavaScript para exibir o conteúdo. Para adicionar um objeto visual a uma janela HTML, você adiciona esse conteúdo ao HTML DOM. As janelas HTML são uma categoria especial de NativeWindow. O host do AIR define uma propriedade `nativeWindow` em janelas HTML que dá acesso à ocorrência subjacente de NativeWindow. Você pode usar essa propriedade para acessar as propriedades, os métodos e os eventos de NativeWindow descritos aqui.

Nota: *O objeto JavaScript Window também tem métodos para gerar scripts da janela que o contém, como `moveTo()` e `close()`. Se houver métodos de sobreposição disponíveis, você poderá usar o método que for mais conveniente.*

Janelas da estrutura do Flex

Quando você cria janelas com a estrutura do Flex, normalmente usa componentes MXML para preencher a janela. Para adicionar um componente do Flex a uma janela, adicione o elemento componente à definição MXML da janela. Também é possível usar o ActionScript para adicionar conteúdo dinamicamente. Os componentes mx:WindowedApplication e mx:Window são projetados como contêineres do Flex e, por isso, podem aceitar componentes do Flex diretamente, enquanto os objetos NativeWindow não. Quando necessário, as propriedades e os métodos de NativeWindow podem ser acessados através dos objetos WindowedApplication e Window usando-se a propriedade `nativeWindow`.

A janela inicial do aplicativo.

A primeira janela do seu aplicativo é criada automaticamente para você pelo AIR. O AIR define as propriedades e o conteúdo da janela usando os parâmetros especificados no elemento `initialWindow` do arquivo de descrição do aplicativo.

Se o conteúdo raiz é um arquivo SWF, o AIR cria uma ocorrência de NativeWindow, carrega o arquivo SWF e o adiciona ao palco da janela. Se o conteúdo raiz é um arquivo HTML, o AIR cria uma janela HTML e carrega o código HTML.

Classes de janela nativa

Adobe AIR 1.0 e posterior

A API de janela nativa contém as seguintes classes:

Pacote	Classes
flash.display	• NativeWindow • NativeWindowInitOptions • NativeWindowDisplayState • NativeWindowResize • NativeWindowSystemChrome • NativeWindowType
flash.events	• NativeWindowBoundsEvent • NativeWindowDisplayStateEvent

Fluxo de eventos de janelas nativas

Adobe AIR 1.0 e posterior

As janelas nativas despacham eventos para notificar os componentes interessados de que uma alteração importante está prestes a ocorrer ou já ocorreu. Muitos eventos relacionados a janelas são despachados em pares. O primeiro evento avisa que uma alteração está prestes a ocorrer. O segundo comunica que a alteração foi feita. É possível cancelar um evento de aviso, mas não um evento de notificação. Esta sequência ilustra o fluxo de eventos que ocorre quando um usuário clica no botão Maximizar de uma janela:

1. O objeto NativeWindow despacha um evento `displayStateChanging`.
2. Se nenhum ouvinte registrado cancelar o evento, a janela será maximizada.
3. O objeto NativeWindow despacha um evento `displayStateChange`.

   Além disso, o objeto NativeWindow despacha eventos de alterações relacionadas feitas no tamanho e na posição de uma janela. A janela não despacha eventos de aviso referentes a essas alterações relacionadas. Os eventos relacionados são:

   a. Um evento `move` é despachado se o canto superior esquerdo da janela foi movido por causa da operação de maximização.
   b. Um evento `resize` é despachado se o tamanho da janela mudou devido à operação de maximização.

   Um objeto NativeWindow despacha uma sequência semelhante de eventos quando maximiza, restaura, fecha, movimenta e redimensiona uma janela.

   Os eventos de aviso são despachados somente quando uma alteração é iniciada através do cromo da janela ou de outro mecanismo controlado pelo sistema operacional. Quando você chama um método de janela para alterar o tamanho, a posição ou o estado de exibição de uma janela, a janela só despacha um evento para comunicar a alteração. Se desejar, você pode despachar um evento de aviso usando o método de janela `dispatchEvent()` e, depois, verificar se o seu evento de aviso foi cancelado antes de prosseguir com a alteração.

   Veja informações detalhadas sobre as classes, os métodos, as propriedades e os eventos da API de janela na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Propriedades que controlam o estilo e o comportamento de janelas nativas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As seguintes propriedades controlam a aparência e o comportamento básicos de uma janela:

- `type`
- `systemChrome`
- `transparent`
- `owner`

Quando cria uma janela, você define essas propriedades no objeto NativeWindowInitOptions passado para o construtor da janela. O AIR lê as propriedades da janela inicial do aplicativo no descritor do aplicativo. (Com exceção da propriedade `type`, que não pode ser definida no descritor do aplicativo e é sempre definida como `normal`.) Não é possível alterar as propriedades após a criação da janela.

Algumas configurações dessas propriedades são mutuamente incompatíveis: `systemChrome` não pode ser definida como `standard` quando `transparent` é `true` ou quando `type` é `lightweight`.

Tipos de janela

Adobe AIR 1.0 e posterior

Os tipos de janela do AIR combinam atributos de cromo e visibilidade do sistema operacional nativo para criar três tipos funcionais de janela. Use as constantes definidas na classe NativeWindowType para fazer referência aos nomes de tipo no código. O AIR oferece os seguintes tipos de janela:

Tipo	Descrição
Normal	Uma janela típica. As janelas normais usam o cromo de estilo em tamanho máximo e são exibidas na barra de tarefas do Windows e no menu de janela do Mac OS X.
Utilitário	Uma paleta de ferramentas. As janelas de utilitário usam uma versão reduzida do cromo do sistema e não são exibidas na barra de tarefas do Windows nem no menu de janela do Mac OS X.
Leve	As janelas leves não têm cromo e não são exibidas na barra de tarefas do Windows nem no menu de janela do Mac OS X. Além disso, as janelas leves não têm o menu Sistema (Alt+espaço) no Windows. As janelas leves são adequadas para as bolhas de notificação e controles como caixas de combinação que abrem uma área de exibição de curta duração. Quando se usa o `tipo` leve, a propriedade `systemChrome` deve ser definida como `none`.

Cromo de janela

Adobe AIR 1.0 e posterior

O cromo de janela é o conjunto de controles que permitem aos usuários manipular uma janela no ambiente da área de trabalho. Os elementos do cromo incluem a barra de título e seus botões, borda e alças de redimensionamento.

Cromo do sistema

É possível definir a propriedade `systemChrome` como `standard` ou `none`. Escolha o cromo do sistema `standard` para dar à sua janela o conjunto de controles padrão criados e estilizados pelo sistema operacional do usuário. Escolha `none` para fornecer o seu próprio cromo para a janela. Use as constantes definidas na classe NativeWindowSystemChrome para fazer referência às configurações de cromo do sistema no código.

O cromo do sistema é gerenciado pelo sistema. Seu aplicativo não tem acesso direto aos controles propriamente ditos, mas pode reagir a eventos despachados quando eles são usados. Quando você usa o cromo padrão para uma janela, a propriedade `transparent` deve ser definida como `false` e a propriedade `type` deve ser `normal` ou `utility`.

Cromo do Flex

Quando você usa os componentes :WindowedApplication ou :Window do Flex, a janela pode utilizar o cromo do sistema ou o cromo fornecido pela estrutura do Flex. Para usar o cromo do Flex, defina a propriedade `systemChrome` usada para criar a janela como `none`. Ao utilizar componentes spark Flex 4 em vez de componentes mx, é necessário especificar a classe da skin para utilizar o Flex chrome. É possível utilizar as skins integradas ou uma outra skin. O exemplo a seguir demonstra como utilizar a classe de skin spark WindowedApplication para fornecer o chrome da janela:

```
<?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>
```

Para obter mais informações, consulte Utilizando o Flex 4: Sobre contêineres de janela do AIR: Controlando o chrome da janela

Cromo personalizado

Se você criar uma janela sem um cromo do sistema, deverá adicionar seus próprios controles de cromo para administrar as interações entre um usuário e a janela. Você também tem liberdade para criar janelas não retangulares transparentes.

Para usar o cromo personalizado com os componentes mx:WindowedApplication ou mx:Window, defina o estilo `showFlexChrome` como `false`. Do contrário, o Flex vai acrescentar seu próprio cromo às suas janelas.

Transparência de janelas

Adobe AIR 1.0 e posterior

Para permitir a mesclagem alfa de uma janela com a área de trabalho ou outras janelas, defina a propriedade `transparent` da janela como `true`. A propriedade `transparent` deve ser definida antes de a janela ser criada e não é possível alterá-la.

Uma janela transparente não tem um plano de fundo padrão. Qualquer área da janela que não contém um objeto desenhado pelo aplicativo fica invisível. Se um objeto exibido tiver uma configuração alfa de menos de 1, qualquer coisa abaixo do objeto ficará visível, inclusive outros objetos de exibição na mesma janela, em outras janelas e na área de trabalho.

As janelas transparentes são úteis quando você deseja criar aplicativos com bordas irregulares ou que “desaparecem” ou parecem ser invisíveis. No entanto, o processamento de grandes áreas com mesclagem alfa pode ser lento, por isso o efeito deve ser usado com cautela.

Importante: *No Linux, eventos de mouse não passam por pixels totalmente transparentes. Você deve evitar criar janelas com áreas grandes totalmente transparentes, já que isso pode bloquear invisivelmente o acesso do usuário a outras janelas ou itens na área de trabalho. No Mac OS X e no Windows, eventos de mouse passam por pixels totalmente transparentes.*

A transparência não pode ser usada com janelas que têm um cromo do sistema. Além disso, o conteúdo SWF e PDF em HTML não pode ser exibido em janelas transparentes. Para obter mais informações, consulte "Considerações ao carregar conteúdo SWF ou PDF em uma página HTML" na página 1040.

A propriedade estática `NativeWindow.supportsTransparency` informa se a transparência de janela está disponível. Quando não há suporte para transparência, o aplicativo é composto com um plano de fundo preto. Nesses casos, qualquer área transparente do aplicativo é exibida como uma área preta opaca. É uma boa prática fornecer um recurso de emergência caso essa propriedade se mostre `false`. Por exemplo, você poderia exibir uma caixa de diálogo de aviso para o usuário ou exibir uma interface de usuário de retangular não transparente.

Vale observar que sempre há suporte para transparência nos sistemas operacionais Mac e Windows. O suporte em sistemas operacionais Linux requer um gerenciador de composição de janela, mas mesmo quando um gerenciador de composição de janela está ativo, a transparência pode estar indisponível devido às opções de exibição do usuário ou configuração de hardware.

Transparência em uma janela de aplicativo MXML

Adobe AIR 1.0 e posterior

Por padrão, o plano de fundo de uma janela MXML é opaco, mesmo que você crie a janela como *transparent*. (Observe o efeito de transparência nos cantos da janela.) Para mostrar um plano de fundo transparente da janela, defina uma cor de plano de fundo e um valor alfa na folha de estilo ou no elemento <mx:Style> contido no arquivo MXML do seu aplicativo. Por exemplo, a seguinte declaração de estilo confere ao plano de fundo uma sombra verde levemente transparente:

```
WindowedApplication
{
    background-alpha:".8";
    background-color:"0x448234";
}
```

Transparência em uma janela de aplicativo HTML

Adobe AIR 1.0 e posterior

Por padrão, o plano de fundo do conteúdo HTML exibido em janelas HTML e objetos HTMLLoader é opaco, mesmo quando a janela que o contém é transparente. Para desativar o plano de fundo padrão exibido com conteúdo HTML, defina a propriedade `paintsDefaultBackground` como `false`. O exemplo abaixo cria um objeto HTMLLoader e desativa o plano de fundo padrão:

```
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.paintsDefaultBackground = false;
```

Este exemplo usa JavaScript para desativar o plano de fundo padrão de uma janela HTML:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Se um elemento do documento HTML define uma cor de plano de fundo, o plano de fundo desse elemento não fica transparente. Não há suporte para definir um valor de transparência parcial (ou opacidade). No entanto, você pode usar um elemento gráfico no formato PNG transparente para uma página ou um elemento de página a fim de obter um efeito visual semelhante.

Propriedade de janelas

Uma janela pode ser *proprietária* de uma ou mais janelas. Essas janelas subordinadas sempre aparecem na frente da janela proprietária, são minimizadas e restauradas juntamente com a janela proprietária e são fechadas quando a janela proprietária é fechada. A propriedade de uma janela não pode ser transferida para outra janela nem removida. Uma janela pode ter apenas uma proprietária, mas pode ser proprietária de qualquer número de outras janelas.

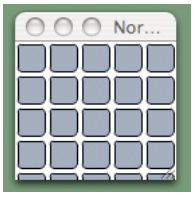
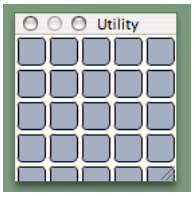
Você pode usar a propriedade de janelas para facilitar o gerenciamento das janelas usadas para paletas de ferramentas e caixas de diálogo. Por exemplo, se você tiver exibido uma caixa de diálogo Salvar associada a uma janela de documento, tornar a janela de documento proprietária da caixa de diálogo manterá automaticamente a caixa de diálogo na frente da janela de documento.

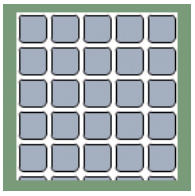
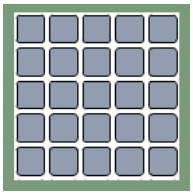
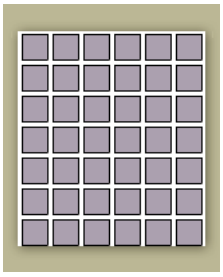
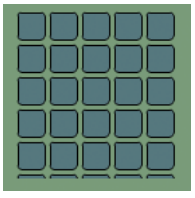
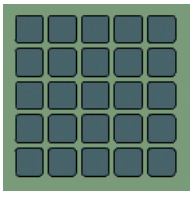
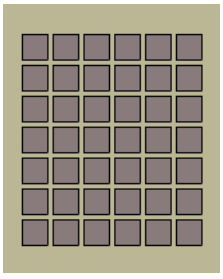
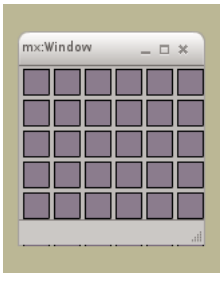
- NativeWindow.owner
- Christian Cantrell: Janelas próprias no AIR 2.6

Um catálogo de janelas visuais

Adobe AIR 1.0 e posterior

A seguinte tabela ilustra os efeitos visuais de diferentes combinações de configurações de propriedades de janela nos sistemas operacionais Mac OS X, Windows e Linux:

Configurações de janela	Mac OS X	Microsoft Windows	Linux*
Tipo: normal SystemChrome: standard Transparente: false			
Tipo: utilitário SystemChrome: standard Transparente: false			

Configurações de janela	Mac OS X	Microsoft Windows	Linux*
Tipo: Qualquer um SystemChrome: none Transparente: false			
Tipo: Qualquer um SystemChrome: none Transparente: true			
mxWindowedApplication ou mx:Window Tipo: Qualquer um SystemChrome: none Transparente: true			

Gerenciador de janela *Ubuntu with Compiz

Nota: *Os seguintes elementos de cromo do sistema não têm suporte no AIR: a barra de ferramentas do OS X, o ícone Proxy do Mac OS X, os ícones na barra de título do Windows e o cromo do sistema alternativo.*

Criação de janelas

Adobe AIR 1.0 e posterior

O AIR cria automaticamente a primeira janela de um aplicativo, mas você pode criar qualquer outra janela de que precise. Para criar uma janela nativa, use o método de construtor NativeWindow.

Para criar uma janela HTML, use o método `createRootWindow()` de HTMLLoader ou, em um documento HTML, chame o método JavaScript `window.open()`. A janela criada é um objeto NativeWindow cuja lista de exibição contém um objeto HTMLLoader. O objeto HTMLLoader interpreta e exibe o conteúdo HTML e JavaScript para a janela. É possível acessar as propriedades o objeto subjacente NativeWindow a partir do JavaScript usando a propriedade `window.nativeWindow`. (Essa propriedade só está acessível ao código em execução na caixa de proteção do aplicativo do AIR.)

Quando uma janela é inicializada—incluindo a janela inicial do aplicativo—você deve considerar criar a janela no estado invisível, carregando o conteúdo ou executando atualizações gráficas e, em seguida, tornando a janela visível. Esta sequência impede a visualização de alterações visuais conflitantes pelos usuários. É possível especificar que a janela inicial do aplicativo deve ser criada no estado invisível, indicando a tag `<visible>false</visible>` no descritor do aplicativo (ou não indicando a tag, já que `false` é o valor padrão). Novas NativeWindows são invisíveis por padrão. Ao criar uma janela HTML com o método HTMLLoader `createRootWindow()`, você pode definir o argumento `visible` para `false`. Chame o método NativeWindow `activate()` ou defina a propriedade `visible` para `true` para tornar a janela visível.

Especificação das propriedades de inicialização de uma janela

Adobe AIR 1.0 e posterior

As propriedades de inicialização de uma janela nativa não podem ser alteradas após a criação da janela da área de trabalho. Entre as propriedades imutáveis e seus valores padrão estão os seguintes:

Propriedade	Valor padrão
systemChrome	standard
tipo	normal
transparent	false
owner	null
maximizable	true
minimizable	true
resizable	true

Defina as propriedades da janela inicial criada pelo AIR no arquivo de descrição do aplicativo. A janela principal de um aplicativo do AIR sempre é do tipo *normal*. (É possível especificar outras propriedades de janela no arquivo de descrição, tais como `visible`, `width` e `height`, mas elas podem ser alteradas a qualquer momento.)

Defina as propriedades de outras janelas nativas e HTML criadas pelo seu aplicativo usando a classe NativeWindowInitOptions. Quando você cria uma janela, deve passar um objeto NativeWindowInitOptions que especifique as propriedades da janela para a função de construtor NativeWindow ou o método `createRootWindow()` de HTMLLoader.

O seguinte código cria um objeto NativeWindowInitOptions para uma janela de utilitário:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.UTILITY
options.transparent = false;
options.resizable = false;
options.maximizable = false;
```

Não é permitido definir *systemChrome* como *standard* quando *transparent* é `true` ou quando `type` é `lightweight`.

Nota: *Não é possível definir as propriedades de inicialização de uma janela criada com a função JavaScript `window.open()`. No entanto, você pode ignorar como elas são criadas implementando sua própria classe HTMLHost. Para obter mais informações, consulte* *"Tratamento de chamadas JavaScript de window.open()" na página 1052.*

Quando criar uma janela com a classe mx:Window do Flex, especifique as propriedades de inicialização no próprio objeto Window, seja na declaração MXML correspondente ou no código que cria a janela. O objeto NativeWindow subjacente só é criado quando você chama o método `open()`. Uma vez aberta uma janela, não é possível alterar essas propriedades de inicialização.

Criação da janela inicial do aplicativo

Adobe AIR 1.0 e posterior

O AIR cria a janela inicial do aplicativo com base nas propriedades especificadas no descritor do aplicativo e carrega o arquivo mencionado no elemento de conteúdo. O elemento de conteúdo deve referenciar um arquivo SWF ou um arquivo HTML.

A janela inicial pode ser a janela principal do seu aplicativo ou pode simplesmente servir para iniciar uma ou mais janelas. Não é necessário torná-la visível.

Criação da janela inicial com o ActionScript

Adobe AIR 1.0 e posterior

Quando você cria um aplicativo do AIR usando o ActionScript, a classe principal do aplicativo deve estender a classe Sprite (ou uma subclasse dela). Esta classe funciona como o principal ponto de entrada do aplicativo.

Quando o seu aplicativo é iniciado, o AIR cria uma janela e uma ocorrência da classe principal e adiciona a ocorrência ao palco da janela. Para acessar a janela, você pode monitorar o evento `addedToStage` e utilizar a propriedade `nativeWindow` do objeto Stage para obter uma referência ao objeto NativeWindow.

Este exemplo ilustra a estrutura básica da classe principal de um aplicativo do AIR criado com o ActionScript:

```
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
        }
    }
}
```

Nota: *Tecnicamente, você PODE acessar a propriedade* `nativeWindow` *na função constructor da classe principal. No entanto, este é um caso especial que se aplica apenas à janela do aplicativo inicial.*

Ao criar um aplicativo no Flash Professional, a classe de documento principal é criada automaticamente se você não criar a sua própria em um arquivo separado do ActionScript. Você pode acessar o objeto NativeWindow para a janela inicial usando a propriedade `nativeWindow`. Por exemplo, o código a seguir ativa a janela principal no estado maximizado (a partir da linha de tempo):

```
import flash.display.NativeWindow;

var mainWindow:NativeWindow = this.stage.nativeWindow;
mainWindow.maximize();
mainWindow.activate();
```

Criação da janela inicial com o Flex

Adobe AIR 1.0 e posterior

Ao criar um aplicativo do AIR com a estrutura do Flex, use mx:WindowedApplication como o elemento raiz do seu principal arquivo MXML. (Você pode usar o componente mx:Application, mas ele não dá suporte a todos os recursos disponíveis no AIR.) O componente WindowedApplication funciona como o ponto de entrada inicial do aplicativo.

Quando você inicia o aplicativo, o AIR cria uma janela nativa, inicializa a estrutura do Flex e adiciona o objeto WindowedApplication ao palco da janela. Ao término da sequência de inicialização, o objeto WindowedApplication despacha um evento `applicationComplete`. Acesse o objeto da janela de área de trabalho com a propriedade `nativeWindow` da ocorrência de WindowedApplication.

O exemplo abaixo cria um componente WindowedApplication simples que define suas coordenadas x e y:

```
<?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>
```

Criação de um objeto NativeWindow

Adobe AIR 1.0 e posterior

Para criar um NativeWindow, passe um objeto NativeWindowInitOptions para o construtor NativeWindow:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.transparent = false;
var newWindow:NativeWindow = new NativeWindow(options);
```

A janela não é exibida até você definir a propriedade `visible` como `true` ou chamar o método `activate()`.

Uma vez criada a janela, você pode inicializar suas propriedades e carregar conteúdo nela usando as técnicas de propriedade de palco e lista de exibição do Flash.

Em quase todos os casos, você deve definir a propriedade de palco `scaleMode` de uma nova janela nativa como `noScale` (use a constante `StageScaleMode.NO_SCALE`). Os modos de dimensionamento do Flash foram desenvolvidos para situações em que o autor do aplicativo não sabe de antemão a proporção do espaço de exibição do aplicativo. Os modos de dimensionamento permitem que o autor escolha o melhor ajuste: recorte o conteúdo, aumente ou compacte o conteúdo ou preencha-o com espaço vazio. Desde que você controle o espaço de exibição no AIR (o quadro da janela), pode dimensionar a janela conforme o conteúdo ou vice-versa, sem comprometer o resultado final.

O modo de dimensionamento de janelas Flex e HTML é definido como `noScale` automaticamente.

Nota: *Para determinar os tamanhos máximo e mínimo de uma janela permitidos no sistema operacional atual, use as seguintes propriedades estáticas da classe NativeWindow:*

```
 var maxOSSize:Point = NativeWindow.systemMaxSize;
var minOSSize:Point = NativeWindow.systemMinSize;
```

Criação de uma janela HTML

Adobe AIR 1.0 e posterior

Para criar uma janela HTML, você pode chamar o método JavaScript `Window.open()` ou o método `createRootWindow()` da classe HTMLLoader do AIR.

O conteúdo HTML em qualquer caixa de proteção de segurança pode usar o método JavaScript `Window.open()` padrão. Se o conteúdo estiver sendo executado fora da caixa de proteção do aplicativo, o método `open()` só poderá ser chamado em resposta a interações do usuário, como cliques do mouse ou pressionamentos de tecla. Quando `open()` é chamado, é criada uma janela com o cromo do sistema para exibir o conteúdo na URL especificada. Por exemplo:

```
newWindow = window.open("xmpl.html", "logWindow", "height=600, width=400, top=10, left=10");
```

Nota: *Você pode estender a classe HTMLHost no ActionScript para personalizar a janela criada com a função JavaScript `window.open()`. Consulte* *"Sobre estender a classe HTMLHost" na página 1044.*

O conteúdo na caixa de proteção de segurança do aplicativo tem acesso ao método de criação de janelas mais eficiente, `HTMLLoader.createRootWindow()`. Com ele, você pode especificar todas as opções de criação de uma nova janela. Por exemplo, o seguinte código JavaScript cria uma janela do tipo leve sem cromo do sistema e com tamanho de 300x400 pixels:

```
 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"));
```

Nota: *Se o conteúdo carregado por uma nova janela estiver fora da caixa de proteção de segurança do aplicativo, o objeto Window não terá as propriedades do AIR: `runtime`, `nativeWindow` ou `htmlLoader`.*

Ao criar uma janela transparente, o conteúdo SWF integrado no HTML carregado nessa janela nem sempre é exibido. Você deve definir o parâmetro `wmode` do objeto ou a tag integrada utilizada para fazer referência ao arquivo SWF para `opaque` ou `transparent`. O valor padrão de `wmode` é `window` e, por padrão, o conteúdo SWF não é exibido em janelas transparentes. O conteúdo PDF não pode ser exibido em janelas transparentes, não importando com o valor `wmode` está definido. (Antes da versão AIR 1.5.2, o conteúdo SWF também não podia ser exibido em janelas transparentes.)

As janelas criadas com o método `createRootWindow()` permanecem independentes da janela de abertura. As propriedades `parent` e `opener` do objeto JavaScript Window são `null`. A janela de abertura pode acessar o objeto Window da nova janela usando a referência a HTMLLoader retornada pela função `createRootWindow()`. No contexto do exemplo anterior, a instrução `newHTMLLoader.window` faria referência ao objeto JavaScript Window da janela criada.

Nota: *A função `createRootWindow()` pode ser chamada no JavaScript e no ActionScript.*

Criação de uma classe mx:Window

Adobe AIR 1.0 e posterior

Para criar uma classe mx:Window, crie um arquivo MXML usando mx:Window como a tag raiz ou chame o construtor de classe Window diretamente.

O exemplo abaixo cria e mostra uma mx:Window chamando o construtor Window:

```
 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);
```

Adição de conteúdo a uma janela

Adobe AIR 1.0 e posterior

O tipo de uma janela do AIR determina a maneira como você adiciona conteúdo a ela. Por exemplo, as linguagens MXML e HTML permitem definir o conteúdo básico da janela declarativamente. É possível incorporar recursos nos arquivos SWF do aplicativo ou carregá-los a partir de arquivos de aplicativo separados. O conteúdo Flex, Flash e HTML pode ser criado dinamicamente e adicionado a uma janela de forma dinâmica.

Quando você carrega conteúdo SWF, ou HTML com JavaScript, deve levar em consideração o modelo de segurança do AIR. Qualquer conteúdo existente na caixa de proteção de segurança do aplicativo, ou seja, conteúdo instalado com seu aplicativo e carregável com o esquema de URL app:, tem privilégios totais para acessar todas as APIs do AIR. Qualquer conteúdo carregado de fora dessa caixa de proteção não pode acessar as APIs do AIR. O conteúdo JavaScript fora da caixa de proteção do aplicativo não pode usar as propriedades `runtime`, `nativeWindow` ou `htmlLoader` do objeto JavaScript Window.

Para permitir código entre scripts seguro, você pode usar uma ponte de caixa de proteção para disponibilizar uma interface limitada entre conteúdo de aplicativo e conteúdo que não é de aplicativo. No conteúdo HTML, também é possível mapear páginas do seu aplicativo para uma caixa de proteção que não seja de aplicativo, de maneira que essa página possa cruzar scripts de conteúdo externo. Consulte “Segurança do AIR” na página 1116.

Carregamento de arquivo ou imagem SWF

Você pode carregar arquivos ou imagens SWF do Flash na lista de exibição de uma janela nativa usando a classe `flash.display.Loader`:

```
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);
        }
    }
}
```

Nota: *Arquivos SWF mais antigos criados com o ActionScript 1 ou 2 compartilham estados globais, como definições de classe, singletons e variáveis globais, se carregados na mesma janela. Se tal arquivo SWF depender de estados globais intocados para funcionar corretamente, não será possível carregá-lo mais de uma vez na mesma janela nem carregá-lo na mesma janela como outro arquivo SWF usando variáveis e definições de classe de sobreposição. Este conteúdo pode ser carregado em janelas separadas.*

Carregamento de conteúdo HTML em um objeto NativeWindow

Para carregar conteúdo HTML em um NativeWindow, você pode adicionar um objeto HTMLLoader ao palco da janela e carregar o conteúdo HTML no HTMLLoader ou criar uma janela que já contenha um objeto HTMLLoader usando o método `HTMLLoader.createRootWindow()`. O seguinte exemplo mostra conteúdo HTML em uma área de exibição de 300 por 500 pixels no palco de uma janela nativa:

```
//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) );
```

Para carregar uma página HTML em um aplicativo do Flex, use o componente HTML do Flex.

O conteúdo SWF em um arquivo HTML não é exibido se a janela utilizar transparência (propriedade `transparent` da janela é `true`) a não ser que o parâmetro `wmode` do objeto ou da tag integrada utilizada para referência ao arquivo SWF esteja definido em `opaque` ou `transparent`. Como o valor padrão de `wmode` é `window`, o conteúdo SWF não é exibido em uma janela transparente. O conteúdo do PDF não é exibido em uma janela transparente, não importando como o valor `wmode` é utilizado.

Além disso, o conteúdo SWF ou PDF não é exibido se o controle HTMLLoader estiver dimensionado ou se a propriedade HTMLLoader `alpha` estiver definida em um valor diferente de 1.0.

Adicionar conteúdo SWF como sobreposição em uma janela HTML

Como as janelas HTML são contidas em uma ocorrência de NativeWindow, é possível adicionar objetos de exibição do Flash tanto acima quanto abaixo da camada HTML na lista de exibição.

Para adicionar um objeto de exibição acima da camada HTML, use o método `addChild()` da propriedade `window.nativeWindow.stage`. O método `addChild()` adiciona conteúdo em camada acima de qualquer conteúdo existente na janela.

Para adicionar um objeto de exibição abaixo da camada HTML, use o método `addChildAt()` da propriedade `window.nativeWindow.stage`, passando um valor de zero para o parâmetro `index`. Colocar um objeto no índice zero movimenta o conteúdo existente, inclusive a exibição HTML, uma camada acima e insere o novo conteúdo na parte inferior. Para que o conteúdo disposto abaixo da página HTML fique visível, defina a propriedade `paintsDefaultBackground` do objeto `HTMLlLoader` como `false`. Além disso, quaisquer elementos da página que têm uma cor de plano de fundo definida não ficarão transparentes. Por exemplo, se você definir uma cor de plano de fundo para o elemento body da página, nenhuma parte dela ficará transparente.

O exemplo a seguir ilustra como adicionar objetos de exibição do Flash na forma de sobreposições e bases a uma página HTML. O exemplo cria dois objetos de forma simples, adiciona um abaixo do conteúdo HTML e um acima. O exemplo também atualiza a posição da forma com base no evento `enterFrame`.

```
 <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>
```

Este exemplo faz uma introdução rudimentar a algumas técnicas avançadas que cruzam os limites entre o JavaScript e o ActionScript no AIR. Se você não estiver familiarizado em utilizar objetos de exibição ActionScript, consulte "Programação de exibição" na página 153 no *Guia do desenvolvedor ActionScript 3.0*.

Exemplo: Criação de uma janela nativa

Adobe AIR 1.0 e posterior

O exemplo abaixo mostra como criar uma janela nativa:

```
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();
}
```

Gerenciamento de janelas

Adobe AIR 1.0 e posterior

Use as propriedades e os métodos da classe NativeWindow para gerenciar a aparência, o comportamento e o ciclo de vida das janelas de área de trabalho.

Nota: *Ao usar a estrutura do Flex, em geral é melhor gerenciar o comportamento da janela usando as classes da estrutura. A maioria das propriedades e métodos NativeWindow podem ser acessados através das classes mx:WindowedApplication e mx:Window classes.*

Obtenção de uma ocorrência de NativeWindow

Adobe AIR 1.0 e posterior

Para manipular uma janela, primeiro você deve obter a ocorrência dela. A ocorrência da janela pode ser obtida em um dos seguintes lugares:

- O construtor nativo de janela usado para criar a janela:

  ```
  var win:NativeWindow = new NativeWindow(initOptions);
  ```

- A propriedade `nativeWindow` do estágio da janela:

  ```
  var win:NativeWindow = stage.nativeWindow;
  ```

- A propriedade `stage` de um objeto de exibição na janela:

  ```
  var win:NativeWindow = displayObject.stage.nativeWindow;
  ```

- A propriedade `target` de um evento de janela nativa despachada pela janela:

  ```
  private function onNativeWindowEvent(event:NativeWindowBoundsEvent):void
  {
      var win:NativeWindow = event.target as NativeWindow;
  }
  ```

- A propriedade `nativeWindow` de uma página HTML exibida na janela:

  ```
  var win:NativeWindow = htmlLoader.window.nativeWindow;
  ```

- As propriedades `activeWindow` e `openedWindows` do objeto NativeApplication:

  ```
  var nativeWin:NativeWindow = NativeApplication.nativeApplication.activeWindow;
  var firstWindow:NativeWindow = NativeApplication.nativeApplication.openedWindows[0];
  ```

  `NativeApplication.nativeApplication.activeWindow` faz referência à janela ativa de um aplicativo (mas retorna `null` se a janela ativa não é uma janela deste aplicativo do AIR). A matriz `NativeApplication.nativeApplication.openedWindows` contém todas as janelas de um aplicativo do AIR que não foram fechadas.

Como os objetos mx:WindowedApplication e Window do Flex são objetos de exibição, você pode fazer referência facilmente à janela do aplicativo em um arquivo MXML usando a propriedade `stage`, da seguinte maneira:

```
 <?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
```

Nota: *Até o componente WindowedApplication ou Window ser adicionado ao palco da janela pela estrutura do Flex, a propriedade `stage` do componente é `null`. Este comportamento é consistente com o do componente Application do Flex, mas significa que não é possível acessar o palco ou a ocorrência de NativeWindow em ouvintes de eventos que ocorrem anteriormente no ciclo de inicialização dos componentes WindowedApplication e Window, como `creationComplete`. É seguro acessar o palco e a ocorrência de NativeWindow quando o evento `applicationComplete` é despachado.*

Ativar, mostrar e ocultar janelas

Adobe AIR 1.0 e posterior

Para ativar uma janela, chame o método `activate()` de NativeWindow. A ativação de uma janela a traz para frente, dá a ela foco do teclado e do mouse e, se necessário, a torna visível mediante a restauração da janela ou a definição da propriedade `visible` como `true`. A ativação de uma janela não altera a ordem de outras janelas do aplicativo. A chamada do método `activate()` faz com que a janela despache um evento `activate`.

Para mostrar uma janela oculta sem ativá-la, defina a propriedade `visible` como `true`. Isso traz a janela para frente, mas não atribui o foco para ela.

Para ocultar uma janela de forma que não seja exibida, defina sua propriedade `visible` como `false`. Ocultar uma janela suprime a exibição tanto da janela quanto de qualquer ícone relacionado na barra de tarefas e, no Mac OS X, a entrada no menu Janelas.

Quando você altera a visibilidade de uma janela, a visibilidade de quaisquer janelas de propriedade dessa janela também é alterada. Por exemplo, se você ocultar uma janela, todas as janelas de sua propriedade também ficarão ocultas.

Nota: *No Mac OS X, não é possível ocultar totalmente uma janela minimizada que tenha um ícone na parte de janela do encaixe. Se a propriedade* `visible` *está definida como* `false` *em uma janela minimizada, o ícone de encaixe da janela continua sendo exibido. Se o usuário clica no ícone, a janela é restaurada para um estado visível e exibida.*

Alteração da ordem de exibição de janelas

Adobe AIR 1.0 e posterior

O AIR oferece diversos métodos para alterar diretamente a ordem de exibição das janelas. É possível mover uma janela para o começo ou o final da ordem de exibição, bem como colocar uma janela em cima de outra ou atrás dela. Ao mesmo tempo, o usuário pode reordenar as janelas ativando-as.

Você pode manter uma janela em frente de outras definindo sua propriedade `alwaysInFront` como `true`. Se mais de uma janela tiver essa configuração, a ordem de exibição dessas janelas será definida entre elas, mas sempre serão organizadas acima das janelas cuja propriedade `alwaysInFront` esteja definida como false.

As janelas que estão no primeiro grupo também são exibidas acima das janelas de outros aplicativos, mesmo quando o aplicativo do AIR não está ativo. Como esse comportamento pode interromper o trabalho do usuário, definir `alwaysInFront` como `true` é algo que só deve ser feito quando necessário e apropriado. Os exemplos de usos justificados incluem:

- Janelas pop-up temporárias de controles, como dicas de ferramentas, listas pop-up, menus personalizados e caixas de combinação. Como essas janelas devem ser fechadas quando perdem o foco, pode-se evitar a dor de cabeça de impedir um usuário de ver outra janela.
- Mensagens de erro e alertas extremamente urgentes. Quando pode ocorrer uma alteração irrevogável se o usuário não responde de imediato, pode ser justificável trazer uma janela de alerta para frente. No entanto, a maioria dos erros e alertas pode ser tratada na ordem de exibição normal das janelas.
- Janelas pop-up de curta duração.

Nota: *O AIR não impõe o uso apropriado da propriedade* `alwaysInFront`*. Entretanto, se o seu aplicativo interrompe o fluxo de trabalho de um usuário, possivelmente será enviado para a lixeira desse mesmo usuário.*

Se uma janela for proprietária de outras, essas outras são sempre ordenadas à sua frente. Se você chamar `orderToFront()` ou definir `alwaysInFront` como `true` em uma janela que seja proprietária de outras, as janelas subordinadas são reordenadas com a janela proprietária em frente às outras janelas, mas as janelas subordinadas ainda são exibidas na frente da proprietária.

Chamar os métodos de ordenação de janelas em janelas subordinadas funciona normalmente entre janelas de propriedade da mesma janela, mas também pode alterar a ordem de todo o grupo de janelas subordinadas em relação a janelas fora desse grupo. Por exemplo, se você chamar `orderToFront()` em uma janela subordinada, essa janela, sua proprietária e quaisquer outras janelas subordinadas à mesma proprietária são movidos para a frente da ordem de exibição de janelas.

A classe NativeWindow oferece estas propriedades e métodos para definir a ordem de exibição de uma janela em relação a outras janelas:

Membro	Descrição
propriedade alwaysInFront	Especifica se a janela é exibida no primeiro grupo de janelas. Em quase todos os casos, `false` é a melhor configuração. Alterar o valor de `false` para `true` traz a janela para frente de todas as janelas (mas não a ativa). Alterar o valor de `true` para `false` coloca a janela atrás das demais janelas do primeiro grupo, mas ainda na frente de outras. Definir a propriedade de uma janela com seu valor atual não altera a ordem de exibição. A configuração `alwaysInFront` não tem nenhum efeito nas janelas de propriedade de outra janela.
orderToFront()	Traz a janela para frente.
orderInFrontOf()	Coloca a janela diretamente à frente de uma determinada janela.
orderToBack()	Envia a janela para trás de outras janelas.
orderBehind()	Coloca a janela diretamente atrás de uma determinada janela.
activate()	Traz a janela para frente (além de torná-la visível e de lhe atribuir foco).

Nota: *Se uma janela está oculta (`visible` é `false`) ou minimizada, chamar os métodos de ordem de exibição não surte nenhum efeito.*

No sistema operacional Linux, diferentes gerenciadores de janela forçam regras diferentes, dependendo da ordem de exibição da janela:

- Em alguns gerenciadores de janela, as janelas do utilitário são sempre exibidas em frente às janelas normais.
- Em alguns gerenciadores de janela, uma janela de tela cheia com `alwaysInFront` definida como `true` é sempre exibida na frente de outras janelas que também tenham `alwaysInFront` definida como `true`.

Fechamento de janelas

Adobe AIR 1.0 e posterior

Para fechar uma janela, use o método `NativeWindow.close()`.

Fechar uma janela descarrega o conteúdo dela; porém, se outros objeto tiverem referências a esse conteúdo, os objetos de conteúdo não serão destruídos. O método `NativeWindow.close()` é executado de forma assíncrona, e o aplicativo que está contido na janela continua a executar durante o processo de fechamento. O método close despacha um evento close quando a operação de fechamento é concluída. O objeto NativeWindow ainda é tecnicamente válido, mas o acesso à maioria das propriedades e dos métodos em uma janela fechada gera IllegalOperationError. Não é possível reabrir uma janela fechada. Verifique a propriedade `closed` de uma janela para testar se a janela foi fechada. Para simplesmente ocultar a exibição de uma janela, defina a propriedade `NativeWindow.visible` como `false`.

Se a propriedade `Nativeapplication.autoExit` é `true`, o padrão, o aplicativo é encerrado ao ser fechada sua última janela.

Quaisquer janelas que tenham uma proprietária são fechadas quando a proprietária é fechada. As janelas que são propriedade de outra não despacham um evento closing e, portanto, não podem impedir seu fechamento. Um evento close é despachado.

Permissão do cancelamento de operações de janela

Adobe AIR 1.0 e posterior

Quando uma janela usa o cromo do sistema, a interação do usuário com ela pode ser cancelada monitorando ou cancelando o comportamento padrão dos eventos apropriados. Por exemplo, quando um usuário clica no botão Fechar do cromo do sistema, o evento `closing` é despachado. Se algum ouvinte registrado chamar o método `preventDefault()` do evento, a janela não será fechada.

Quando uma janela não usa o cromo do sistema, os eventos de notificação referentes às alterações pretendidas não são despachados automaticamente antes de a alteração ser feita. Por isso, se você chamar os métodos para fechar uma janela ou se alterar o estado dela ou definir qualquer uma das propriedades de limites de janela, a alteração não poderá ser cancelada. Para notificar os componentes do seu aplicativo antes que seja feita uma alteração de janela, a lógica do aplicativo poderá despachar o evento de notificação relevante usando o método `dispatchEvent()` da janela.

Por exemplo, a seguinte lógica implementa um manipulador de eventos cancelável para o botão Fechar de uma janela:

```
public function onCloseCommand(event:MouseEvent):void{
    var closingEvent:Event = new Event(Event.CLOSING,true,true);
    dispatchEvent(closing);
    if(!closingEvent.isDefaultPrevented()){
        win.close();
    }
}
```

O método `dispatchEvent()` retorna `false` se o método `preventDefault()` do evento é chamado por um ouvinte. No entanto, ele também pode retornar `false` por outros motivos, portanto é melhor usar o método `isDefaultPrevented()` explicitamente para testar se a alteração deve ser cancelada.

Maximização, minimização e restauração de uma janela

Adobe AIR 1.0 e posterior

Para maximizar a janela, use o método `maximize()` da classe NativeWindow.

```
myWindow.maximize();
```

Para minimizar a janela, use o método `minimize()` de NativeWindow.

```
myWindow.minimize();
```

Para restaurar a janela (isto é, colocá-la novamente no tamanho em que estava antes de ser minimizada ou maximizada), use o método `restore()` de NativeWindow.

```
myWindow.restore();
```

Uma janela que tenha uma proprietária é minimizada e restaurada quando a janela proprietária é minimizada ou restaurada. Nenhum evento é despachado pela janela subordinada quando é minimizada porque sua proprietária está minimizada.

Nota: *O comportamento resultante da maximização de uma janela do AIR é diferente do comportamento padrão do Mac OS X. Em vez de alternar entre um tamanho "padrão" definido pelo aplicativo e o último tamanho definido pelo usuário, as janelas do AIR alternam entre o último tamanho definido pelo aplicativo ou pelo usuário e a área útil total da tela.*

No sistema operacional Linux, diferentes gerenciadores de janela forçam regras diferentes com relação a configuração do estado de exibição da janela:

- Em alguns gerenciadores de janela, não é possível maximizar as janelas dos utilitários.
- Se um tamanho máximo for definido para a janela, algumas janelas não permitirão a maximização. Alguns outros gerenciadores de janela definem o estado de exibição como maximizado, mas não redimensionam a janela. Em qualquer um dos casos, não é despachado nenhum evento de alteração do estado de exibição.
- Alguns gerenciadores de janela não cumprem as configurações `maximizable` ou `minimizable` de janela.

Nota: *No Linux, as propriedades da janela são alteradas de forma assíncrona. Se você alterar o estado de exibição em uma linha do programa e ler os valores na linha seguinte, o valor lido ainda refletirá a configuração antiga. Em todas as plataformas, o objeto NativeWindow despacha o evento* `displayStateChange` *quando o estado de exibição muda. Se você precisar realizar algumas ações com base no novo estado da janela, sempre faça isso em um manipulador de eventos* `displayStateChange`*. Consulte "Monitorando eventos de janela" na página 945.*

Exemplo: minimização, maximização, restauração e fechamento de uma janela

Adobe AIR 1.0 e posterior

O seguinte aplicativo MXML demonstra os métodos `maximize()`, `minimize()`, `restore()` e `close()` de janela:

```
<?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>
```

O seguinte exemplo do ActionScript para Flash cria quatro campos de texto clicáveis que acionam os métodos `minimize()`, `maximize()`, `restore()` e `close()` de NativeWindow:

```
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();
        }
    }
}
```

Redimensionamento e movimentação de uma janela

Adobe AIR 1.0 e posterior

Quando uma janela usa o cromo do sistema, o cromo oferece controles de arrastar que permitem redimensionar a janela e movimentá-la na área de trabalho. Se uma janela não usa o cromo do sistema, adicione seus próprios controles para que o usuário possa redimensionar e movimentar a janela.

Nota: *Para redimensionar ou movimentar uma janela, primeiro você deve obter uma referência à ocorrência de NativeWindow. Para mais informações sobre como obter uma referência à janela, consulte "Obtenção de uma ocorrência de NativeWindow" na página 935.*

Redimensionamento de uma janela

Para permitir que um usuário redimensione uma janela de forma interativa, use o método NativeWindow`startResize()`. Quando esse método é chamado a partir de um evento `mouseDown`, a operação de redimensionamento é orientada pelo mouse e executada quando o sistema operacional recebe um evento `mouseUp`. Quando chama `startResize()`, você passa um argumento que especifica a borda ou o canto a partir do qual a janela deve ser redimensionada.

Para definir o tamanho das janelas de forma programática, defina as propriedades `width`, `height` ou `bounds` da janela nas dimensões desejadas. Quando você define os limites, o tamanho da janela e a posição podem ser alterados ao mesmo tempo., No entanto, não há garantias da ordem de exibição das alterações. Alguns gerenciadores de janela do Linux não permitem que janelas se estendam além dos limites da tela da área de trabalho. Nesses casos, o tamanho final da janela pode ser limitado devido à ordem em que as propriedades são definidas, apesar de as alterações resultarem em uma janela legal. Por exemplo, se você alterar a altura e a posição y de uma janela perto da parte inferior da tela, a alteração na altura total pode não ocorrer quando for aplicada antes da alteração na posição y.

Nota: *No Linux, as propriedades da janela são alteradas de forma assíncrona. Se você redimensionar uma janela em uma linha do seu programa, e ler as dimensões na próxima, elas ainda vão refletir as configurações antigas. Em todas as plataformas, o objeto NativeWindow despacha o evento `resize` quando a janela é redimensionada. Se precisar realizar alguma ação, como dispor controles na janela, com base no novo tamanho ou estado da janela, sempre faça isso em um manipulador de eventos `resize`. Consulte "Monitorando eventos de janela" na página 945.*

O modo de dimensionamento do palco determina o comportamento do palco da janela e de seu conteúdo quando uma janela é redimensionada. Lembre-se de que os modos de dimensionamento de palco devem ser usados em situações em que o aplicativo não tem controle sobre o tamanho ou a proporção de seu espaço de exibição, como é o caso de um navegador da Web. Em geral, os melhores resultados são obtidos quando você define a propriedade `scaleMode` do palco como `StageScaleMode.NO_SCALE`. Para que o conteúdo da janela seja dimensionado, também é possível definir os parâmetros do conteúdo `scaleX` e `scaleY` em resposta às alterações nos limites da janela.

Movimentação de uma janela

Para mover uma janela sem redimensioná-la, utilize o método NativeWindow `startMove()`. Como acontece com o método `startResize()`, quando você chama o método `startMove()` a partir de um evento `mouseDown`, o processo de movimentação é orientado pelo mouse e executado quando o sistema operacional recebe um evento `mouseUp`.

Para obter mais informações, sobre os métodos `startResize()` e `startMove()`, consulte a Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Para mover uma janela em termos programáticos, defina as propriedades `x`, `y` ou `bounds` da janela até a posição desejada. Quando você define os limites, o tamanho da janela e a posição podem ser alterados ao mesmo tempo.

Nota: *No Linux, as propriedades da janela são alteradas de forma assíncrona. Se você mover uma janela em uma linha do programa e ler a posição na linha seguinte, o valor lido ainda refletirá a configuração antiga. Em todas as plataformas, o objeto NativeWindow despacha o evento* `move` *quando a posição muda. Se você precisar realizar algumas ações com base na nova posição da janela, sempre faça isso em um manipulador de eventos* `move`*. Consulte "Monitorando eventos de janela" na página 945.*

Exemplo: redimensionamento e movimentação de janelas

Adobe AIR 1.0 e posterior

O exemplo abaixo mostra como iniciar operações de redimensionamento e movimentação em uma janela:

```
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();
        }
    }
}
```

Monitorando eventos de janela

Adobe AIR 1.0 e posterior

Para monitorar os eventos despachados por uma janela, registre um ouvinte na ocorrência da janela. Por exemplo, para monitorar o evento closing, registre um ouvinte na janela, da seguinte maneira:

```
myWindow.addEventListener(Event.CLOSING, onClosingEvent);
```

Quando um evento é despachado, a propriedade `target` faz referência à janela que o está enviando.

A maioria dos eventos de janela tem duas mensagens relacionadas. A primeira mensagem sinaliza que uma alteração de janela é iminente (e pode ser cancelada), enquanto a segunda indica que a alteração foi executada. Por exemplo, quando um usuário clica no botão Fechar de uma janela, a mensagem do evento closing é despachada. Se nenhum ouvinte cancelar o evento, a janela será fechada e o evento close será despachado para qualquer ouvinte.

Normalmente, os eventos de aviso, como `closing`, só são despachados quando o cromo do sistema foi usado para acionar um evento. Se você chamar o método de janela `close()`, por exemplo, o evento `closing` não será despachado automaticamente (será despachado apenas o evento `close`). No entanto, você pode construir um objeto de evento closing e despachá-lo usando o método de janela `dispatchEvent()`.

Os eventos de janela que despacham um objeto Event são:

Evento	Descrição
activate	Despachado quando a janela recebe foco.

Evento	Descrição
deactivate	Despachado quando a janela perde foco.
closing	Despachado quando a janela está prestes a fechar. Este evento só ocorre automaticamente quando o botão Fechar do cromo do sistema é pressionado ou, no Mac OS X, quando é chamado o comando Quit.
close	Despachado quando a janela foi fechada.

Os eventos de janela que despacham um objeto NativeWindowBoundsEvent são:

Evento	Descrição
moving	Despachado imediatamente depois que o canto superior esquerdo da janela muda de posição, seja como resultado da movimentação, do redimensionamento ou da alteração do estado de exibição da janela.
move	Despachado depois que a posição do canto superior esquerdo é alterada.
resizing	Despachado imediatamente antes de a largura ou a altura da janela ser alterada, seja como resultado do redimensionamento ou de uma alteração no estado de exibição.
resize	Despachado após uma alteração no tamanho da janela.

Para eventos NativeWindowBoundsEvent, você pode usar as propriedades `beforeBounds` e `afterBounds` para determinar os limites da janela antes e depois da alteração iminente ou concluída.

Os eventos de janela que despacham um objeto NativeWindowDisplayStateEvent são:

Evento	Descrição
displayStateChanging	Despachado imediatamente antes de uma alteração no estado de exibição da janela.
displayStateChange	Despachado depois que o estado de exibição da janela foi alterado.

Para eventos NativeWindowDisplayStateEvent , você pode usar as propriedades `beforeDisplayState` e `afterDisplayState` para determinar o estado de exibição da janela antes e depois da alteração iminente ou concluída.

Em alguns gerenciadores de janela do Linux, um evento de alteração do estado de exibição não é despachado quando uma janela com uma configuração de tamanho máximo é maximizada. (A janela é definida como estado de exibição máximo, mas não é redimensionada).

Exibição de janelas em tela cheia

Adobe AIR 1.0 e posterior

Definir a propriedade `displayState` da classe Stage como `StageDisplayState.FULL_SCREEN_INTERACTIVE` coloca a janela em modo de tela cheia, que *permite* entrada pelo teclado. (No conteúdo SWF em execução em um navegador, a entrada pelo teclado não é permitida). Para sair do modo de tela cheia, o usuário deve pressionar a tecla Escape.

Nota: *Alguns gerenciadores de janela do Linux não alterarão as dimensões de janela para encher a tela, caso um tamanho máximo seja definido para a janela (mas removerão o cromo do sistema da janela).*

Por exemplo, o seguinte código do Flex define um aplicativo simples do AIR que configura um terminal simples para tela cheia:

```
 <?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>
```

O seguinte exemplo do ActionScript para Flash simula um terminal de texto simples em tela cheia:

```
    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);
        }
    }
```

Capítulo 53: Exibição de telas no AIR

Adobe AIR 1.0 e posterior

Use a classe Screen do Adobe® AIR® para acessar informações sobre as telas de exibição anexadas a um computador ou dispositivo.

Mais tópicos da Ajuda

flash.display.Screen

Princípios básicos das telas de exibição no AIR

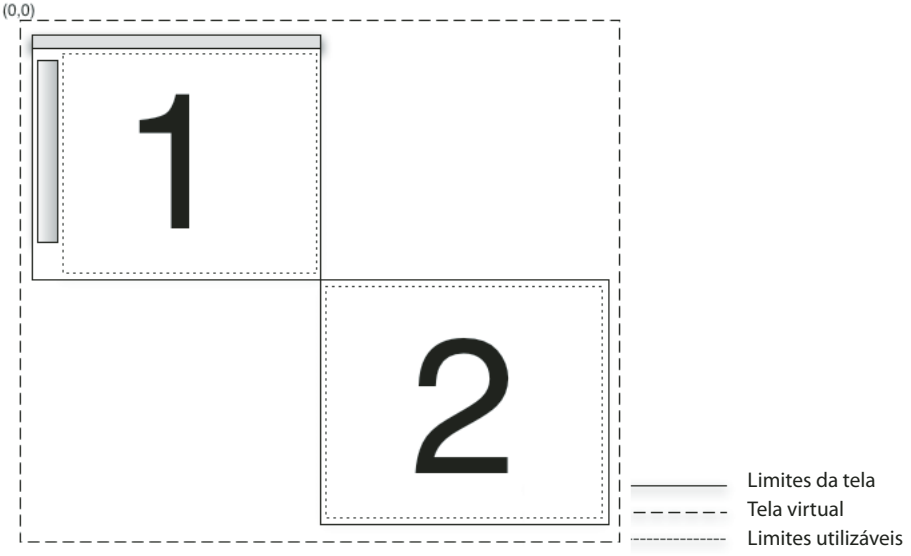
Adobe AIR 1.0 e posterior

- Medida da área de trabalho virtual (Flex)
- Medida da área de trabalho virtual (Flash)

A API de tela contém uma classe única, Screen, que oferece membros estáticos para obter informações de tela do sistema e membros de ocorrência para descrever uma tela específica.

Um sistema de computador pode ter vários monitores ou exibições anexadas, que podem corresponder a diversas telas da área de trabalho organizadas em um espaço virtual. A classe Screen do AIR fornece informações sobre as telas, sua organização relativa e seu espaço utilizável. Se mais de um monitor for mapeado para a mesma tela, haverá somente uma tela. Se o tamanho da tela é maior que a área de exibição do monitor, não há como determinar que parte da tela está visível atualmente

A tela representa uma área de exibição independente da área de trabalho. As telas são descritas como retângulos na área de trabalho virtual. O canto superior esquerdo da tela designado como exibição principal é a origem do sistema de coordenadas da área de trabalho virtual. Todos os valores usados para descrever a tela são fornecidos em pixels.



Nessa organização de telas, há duas telas na área de trabalho virtual. As coordenadas do canto superior esquerdo da tela principal (nº 1) são sempre (0,0). Se a organização da tela for alterada para designar a tela nº 2 como tela principal, as coordenadas da tela nº 1 se tornarão negativas. Barras de menu, barras de tarefa e encaixes são excluídos ao reportar limites utilizáveis de uma tela.

Para obter informações mais detalhadas sobre a função da classe API, métodos, propriedades e eventos, consulte a Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Enumeração das telas

Adobe AIR 1.0 e posterior

Você pode enumerar as telas da área de trabalho virtual com os seguintes métodos e propriedades de tela:

Método ou Propriedade	Descrição
Screen.screens	Oferece uma matriz de objetos Screen que descreve as telas disponíveis. A ordem da matriz não é significativa.
Screen.mainScreen	Oferece o objeto Screen da tela principal. No Mac OS X, a tela principal é a tela que exibe a barra de menu. No Windows, a tela principal é a tela primária designada pelo sistema.
Screen.getScreensForRectangle()	Oferece uma matriz de objetos Screen que descreve as telas intersectadas pelo retângulo determinado. O retângulo passado para este método está em coordenadas de pixel na área de trabalho virtual. Se nenhuma tela intersectar o retângulo, a matriz ficará vazia. Você pode usar esse método para descobrir em que telas a janela é exibida.

Não salve os valores retornados pelos métodos e propriedades da classe Screen. O usuário ou sistema operacional pode alterar as telas disponíveis e a respectiva disposição a qualquer momento.

O exemplo a seguir usa a API da tela para mover uma janela entre várias telas em resposta ao pressionamento das teclas de seta. Para mover a janela para a próxima tela, o exemplo obtém a matriz `screens` e a classifica verticalmente ou horizontalmente (dependendo da tecla de seta pressionada). Em seguida, o código passa pela matriz classificada, comparando cada tela com as coordenadas da tela atual. Para identificar a tela atual da janela, o exemplo chama `Screen.getScreensForRectangle()`, passando nos limites da janela.

```
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;
    }
  }
}
```

Capítulo 54: Impressão

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os tempos de execução do Flash (como o Adobe® Flash® Player e o Adobe® AIR™) podem se comunicar com a interface de impressão de um sistema operacional para que você consiga passar as páginas para o spooler de impressão. Cada página enviado ao spooler pode ter conteúdo visível, dinâmico ou fora da tela do usuário, incluindo valores de banco de dados e texto dinâmico. Além disso, as propriedades da classe flash.printing.PrintJob contêm valores com base nas configurações da impressora do usuário, para que você possa formatar as páginas adequadamente.

O conteúdo a seguir detalha estratégias para uso dos métodos e propriedades da classe flash.printing.PrintJob para criar um trabalho de impressão, ler as configurações de impressão de um usuário e ajustar um trabalho de impressão com base no comentário do aplicativo Flash e no sistema operacional do usuário.

Noções básicas de impressão

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No ActionScript 3.0, você usa a classe PrintJob para criar instantâneos de conteúdo de exibição a serem convertidos em uma representação de tinta e papel em uma impressão. Sob alguns aspectos, configurar o conteúdo para impressão é o mesmo que configurá-lo para exibição na tela: você posiciona e classifica os elementos por tamanho para criar o layout desejado. No entanto, a impressão possui características únicas que a tornam diferente do layout de tela. Por exemplo, a resolução usada por impressoras é diferente da resolução de monitores de computador. O conteúdo de uma tela do computador é dinâmico e pode ser alterado, enquanto o conteúdo impresso é basicamente estático, e no planejamento da impressão, as restrições de tamanho da página fixa e a possibilidade de impressão de várias páginas precisam ser consideradas.

Embora essas diferenças pareçam óbvias, é importante tê-las em mente ao configurar a impressão com o ActionScript. A impressão exata depende de uma combinação de valores especificados por você e das características da impressora do usuário. A classe PrintJob contém propriedades que permitem determinar as características importantes da impressora do usuário.

Conceitos e termos importantes

A lista de referência a seguir contém termos importantes relacionados à impressão:

Spooler Uma parte do sistema operacional ou do software do driver da impressora que mantém o controle das páginas que estão aguardando para serem impressas e as envia para a impressora quando ela estiver disponível.

Orientação de página A rotação do conteúdo impresso em relação ao papel, horizontal (paisagem) ou vertical (retrato).

Trabalho de impressão A página ou o conjunto de páginas que compõem uma única impressão.

Impressão de uma página

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você usa uma ocorrência da classe PrintJob para manipular a impressão. Para imprimir uma página básica por meio do Flash Player ou do AIR, use estas quatro instruções em sequência:

- `new PrintJob()`: Cria uma nova ocorrência do trabalho de impressão com o nome especificado.
- `PrintJob.start()`: Inicia o processo de impressão do sistema operacional chamando a caixa de diálogo de impressão do usuário e preenche as propriedades somente leitura do trabalho de impressão.
- `PrintJob.addPage()`: Contém os detalhes sobre o conteúdo do trabalho de impressão, incluindo o objeto Sprite (e todos os filhos que ele contém), o tamanho da área de impressão e se a impressora imprimirá a imagem como um vetor ou um bitmap. Você pode usar chamadas sucessivas para `addPage()` para imprimir várias entidades gráficas em várias páginas.
- `PrintJob.send()`: Envia as páginas para a impressora do sistema operacional.

Portanto, por exemplo, um script simples de trabalho de impressão é (inclusive as instruções `package`, `import` e `class` para compilação):

```
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();
        }
    }
}
```

Nota: *Esse exemplo tem o objetivo de mostrar os elementos básicos de um script de trabalho de impressão e não contém nenhuma manipulação de erros. Para criar um script que responda corretamente a um usuário que cancela um trabalho de impressão, consulte* *"Trabalho com exceções e retornos" na página 955.*

Para limpar as propriedades de um objeto PrintJob por qualquer motivo, defina a variável PrintJob como `null` (como em `myPrintJob = null`).

Tarefas do tempo de execução e impressão de sistema do Flash

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Devido aos tempos de execução do Flash enviarem páginas para a interface de impressão do sistema operacional, observe as tarefas gerenciadas pelos tempos de execução do Flash e tarefas gerenciadas pela interface de impressão do próprio sistema operacional. Os aplicativos Flash podem iniciar um trabalho de impressão, ler algumas configurações de página de uma impressora, transmitir o conteúdo de um trabalho de impressão para o sistema operacional e verificar se o usuário ou o sistema cancelou um trabalho de impressão. Outros processos, como exibir caixas de diálogo específicas à impressora, cancelar um trabalho de impressão no spool ou relatar o status da impressora, são todos manipulados pelo sistema operacional. Os tempos de execução do Flash podem responder se há um problema ao iniciar ou formatar um trabalho de impressão, mas podem relatar apenas determinadas propriedades ou condições da interface de impressão do sistema operacional. Como desenvolvedor, seu código precisa responder a essas propriedades ou condições.

Trabalho com exceções e retornos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Verifique se o método `PrintJob.start()` retorna `true` antes de executar as chamadas `addPage()` e `send()`, no caso do usuário ter cancelado o trabalho de impressão. Uma maneira simples de verificar se esses métodos foram cancelados antes de continuar é delimitá-los em uma instrução `if`, da seguinte maneira:

```
if (myPrintJob.start())
{
    // addPage() and send() statements here
}
```

Se `PrintJob.start()` for `true`, o Print selecionado pelo usuário (ou um aplicativo Flash, como o Flash Player ou o AIR, iniciou um comando Print). Portanto, os métodos `addPage()` e `send()` podem ser chamados.

Além disso, para ajudar a gerenciar o processo de impressão, os tempos de execução do Flash emitem exceções para o método `PrintJob.addPage()` para que seja possível capturar os erros e fornecer informações e opções ao usuário. Se um método `PrintJob.addPage()` falhar, você também poderá chamar outra função ou parar o trabalho de impressão atual. Você captura essas exceções incorporando as chamadas de `addPage()` dentro de uma instrução `try..catch`, conforme no exemplo a seguir. No exemplo, `[params]` é um alocador de espaço para os parâmetros que especificam o conteúdo real a ser impresso:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error,
    }
    myPrintJob.send();
}
```

Depois que o trabalho de impressão for iniciado, você pode adicionar o conteúdo usando `PrintJob.addPage()` e verificar se isso gera uma exceção (por exemplo, se o usuário cancelou o trabalho de impressão). Se for gerada uma exceção, você poderá adicionar lógica à instrução `catch` para fornecer informações e opções ao usuário (ou ao tempo de execução do Flash) ou poderá parar o trabalho de impressão atual. Se você adicionar a página com êxito, poderá continuar a enviar as páginas à impressora usando `PrintJob.send()`.

Se o tempo de execução do Flash encontrar um problema quando enviar o trabalho de impressão para a impressora (por exemplo, se a impressora estiver offline), você também poderá capturar essa exceção e apresentar informações ou mais opções (por exemplo, exibir o texto da mensagem ou fornecer um alerta dentro de uma animação). Por exemplo, você pode atribuir texto novo a um campo de texto em uma instrução `if..else`, conforme mostrado no código a seguir:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error.
    }
    myPrintJob.send();
}
else
{
    myAlert.text = "Print job canceled";
}
```

Para obter um exemplo que funciona, consulte “Exemplo de impressão: escala, corte e resposta” na página 964.

Trabalho com propriedades da página

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando o usuário clica em OK na caixa de diálogo Imprimir e `PrintJob.start()` retorna `true`, ele pode acessar as propriedades definidas pelas configurações da impressora. Essas configurações incluem a largura e a altura do papel (`pageHeight` e `pageWidth`), e a orientação do conteúdo no papel. Como essas são configurações da impressora, não controladas pelo tempo de execução do Flash, você não pode alterá-las. Mas pode usá-las para alinhar o conteúdo enviado para a impressora para que corresponda às configurações atuais. Para obter mais informações, consulte “Configuração de tamanho, escala e orientação” na página 958.

Configuração da renderização de vetores ou de bitmaps

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode definir manualmente o trabalho de impressão para armazenar no spool cada página, como gráficos de vetor ou uma imagem de bitmap. Em alguns casos, a impressão de vetores produz um arquivo de spool menor e uma imagem melhor do que a impressão de bitmap. No entanto, se o conteúdo incluir uma imagem de bitmap, e você desejar preservar qualquer transparência alfa ou efeitos de cor, deverá imprimir a página como uma imagem de bitmap. Além disso, uma impressora não PostScript converte automaticamente todos os gráficos de vetor em imagens de bitmap.

É possível especificar a impressão de bitmap, indicando o objeto PrintJobOptions como terceiro parâmetro de `PrintJob.addPage()`.

Para o Flash Player e AIR anterior ao AIR 2, defina o parâmetro `printAsBitmap` do objeto PrintJobOptions para `true`, conforme o exemplo a seguir:

```
var options:PrintJobOptions = new PrintJobOptions();
options.printAsBitmap = true;
myPrintJob.addPage(mySprite, null, options);
```

Se você não especificar um valor para o terceiro parâmetro, o trabalho de impressão usará o padrão, que é a impressão de vetor.

Para o AIR 2 e posterior, utilize a propriedade `printMethod` do objeto PrintJobOptions para especificar o método de impressão. Essa propriedade aceita três valores, definidos como constantes na classe PrintMethod:

- `PrintMethod.AUTO`: Automaticamente selecione o melhor método de impressão com base no conteúdo impresso. Por exemplo, se a página consiste em texto, o método de impressão vetorial é selecionado. No entanto, se uma imagem de marca d'água com transparência alfa for sobreposta no texto, a impressão de bitmap é selecionada para manter a transparência.
- `PrintMethod.BITMAP`: Força a impressão bitmap, imprimindo independentemente do conteúdo
- `PrintMethod.BITMAP`: Força a impressão vetor, imprimindo independentemente do conteúdo

Controle de tempo de instruções de trabalho de impressão

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O ActionScript 3.0 não restringe o objeto PrintJob a um único quadro (como faziam as versões anteriores do ActionScript). No entanto, como o sistema operacional exibe informações do status de impressão para o usuário depois que ele clicou no botão OK na caixa de diálogo Imprimir, você deve chamar `PrintJob.addPage()` e `PrintJob.send()` assim que possível para enviar as páginas para o spooler. Um atraso que atinja o quadro que contém a chamada do `PrintJob.send()` atrasará o processo de impressão.

No ActionScript 3.0, há um tempo limite de script de 15 segundos. Portanto, o tempo entre cada instrução principal em uma sequência de trabalho de impressão não pode exceder 15 segundos. Em outras palavras, o tempo limite de 15 segundos de script se aplica aos seguintes intervalos:

- Entre `PrintJob.start()` e o primeiro `PrintJob.addPage()`
- Entre `PrintJob.addPage()` e o próximo `PrintJob.addPage()`
- Entre o último `PrintJob.addPage()` e `PrintJob.send()`

Se qualquer um desses intervalos se estender por mais de 15 segundos, a chamada seguinte para o `PrintJob.start()`, na ocorrência PrintJob, retornará `false` e o seguinte `PrintJob.addPage()` na ocorrência PrintJob fará com que o Flash Player ou o AIR emitam uma exceção de tempo de execução.

Configuração de tamanho, escala e orientação

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A seção "Impressão de uma página" na página 954 fornece detalhes das etapas de um trabalho de impressão básico, em que a saída reflete diretamente o equivalente impresso do tamanho e posição da tela da entidade gráfica especificada. No entanto as impressoras usam resoluções diferentes para impressão e podem ter configurações que afetam contrariamente a aparência da entidade gráfica impressa.

Os tempos de execução do Flash podem ler configurações de impressão de um sistema operacional, mas observe que essas propriedades são somente leitura: embora você possa responder a seus valores, não pode defini-los. Portanto, por exemplo, você pode localizar a configuração de tamanho da página da impressora e ajustar seu conteúdo para se adequar ao tamanho. Também é possível determinar as configurações de margem e de orientação da página da impressora. Para responder às configurações da impressora, especifique uma área de impressão, ajuste a diferença entre a resolução da tela e as medidas de ponto da impressora ou transforme o conteúdo para atender às configurações de tamanho ou de orientação da impressora do usuário.

Uso de retângulos para a área de impressão

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `PrintJob.addPage()` permite especificar a região de uma entidade gráfica que você deseja que seja impressa. O segundo parâmetro, `printArea`, está na forma de um objeto Rectangle. Há três opções para fornecer um valor para esse parâmetro:

- Crie um objeto Rectangle com propriedades específicas e, em seguida, use esse retângulo na chamada de `addPage()`, conforme no exemplo a seguir:

  ```
  private var rect1:Rectangle = new Rectangle(0, 0, 400, 200);
  myPrintJob.addPage(sheet, rect1);
  ```

- Se você ainda não tiver especificado um objeto Rectangle, poderá fazer isso dentro da própria chamada, como no exemplo a seguir:

  ```
  myPrintJob.addPage(sheet, new Rectangle(0, 0, 100, 100));
  ```

- Se você planejar fornecer valores para o terceiro parâmetro na chamada de `addPage()`, mas não desejar especificar um retângulo, poderá usar `null` para o segundo parâmetro, como no exemplo a seguir:

  ```
  myPrintJob.addPage(sheet, null, options);
  ```

Comparação de pontos e pixels

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A largura e a altura do retângulo são valores em pixels. Uma impressora usa pontos como unidades de medida de impressão. Os pontos têm tamanho físico fixo (1/72 polegadas), mas o tamanho de um pixel na tela depende da resolução da tela específica. A taxa de conversão entre pixels e pontos depende das configurações da impressora e se a entidade gráfica está dimensionada. Um sprite não dimensionado, com 72 pixels de largura, terá uma polegada de largura na impressão, com um ponto igual a um pixel, independentemente da resolução da tela.

Você pode usar as equivalências a seguir para converter polegadas ou centímetros em twips (1/20 de um ponto) ou pontos:

- 1 ponto = 1/72 pol = 20 twips

- 1 pol = 72 pontos = 1440 twips
- 1 centímetro = 567 twips

Se você omitir o parâmetro `printArea`, ou se ele for passado incorretamente, a área total da entidade gráfica será impressa.

Escala

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para dimensionar um objeto Sprite antes de imprimi-lo, defina as propriedades da escala (consulte "Manipulação do tamanho e dimensionamento de objetos" na página 182) antes de chamar o método `PrintJob.addPage()` e defina-as novamente como seus valores originais após a impressão. A escala de um objeto Sprite não tem nenhuma relação com a propriedade `printArea`. Em outras palavras, se você especificar uma área de impressão de 50 x 50 pixels, serão impressos 2500 pixels. Se você dimensionar o objeto Sprite, serão impressos os mesmos 2500 pixels, mas o objeto Sprite será impresso no tamanho dimensionado.

Para obter um exemplo, consulte "Exemplo de impressão: escala, corte e resposta" na página 964.

Impressão de orientação paisagem ou retrato

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Como o Flash Player e o AIR podem detectar as configurações de orientação, você pode criar lógica no ActionScript para ajustar o tamanho ou a rotação do conteúdo em resposta às configurações da impressora, conforme ilustrado no exemplo a seguir:

```
if (myPrintJob.orientation == PrintJobOrientation.LANDSCAPE)
{
    mySprite.rotation = 90;
}
```

Nota: *Se você planeja ler a configuração do sistema para obter a orientação do conteúdo no papel, lembre-se de importar a classe PrintJobOrientation. A classe PrintJobOrientation fornece valores de constante que definem a orientação do conteúdo na página. Importe a classe usando a seguinte instrução:*

```
import flash.printing.PrintJobOrientation;
```

Resposta à altura e largura da página

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Usando uma estratégia semelhante à manipulação das configurações de orientação da impressora, você pode ler as configurações de altura e largura da página e responder a elas incorporando alguma lógica em uma instrução `if`. O código a seguir mostra um exemplo:

```
if (mySprite.height > myPrintJob.pageHeight)
{
    mySprite.scaleY = .75;
}
```

Além disso, as configurações da margem de uma página podem ser determinadas comparando as dimensões da página e do papel, conforme ilustrado no exemplo a seguir:

```
margin_height = (myPrintJob.paperHeight - myPrintJob.pageHeight) / 2;
margin_width = (myPrintJob.paperWidth - myPrintJob.pageWidth) / 2;
```

Técnicas avançadas de impressão

Adobe AIR 2 e posterior

Desde o Adobe AIR 2, a classe PrintJob possui propriedades adicionais, métodos e suporte a três classes adicionais: PrintUIOptions, PaperSize e PrintMethod. Essas mudanças permitem adicionar novos fluxos de trabalho de impressora e permite que os autores tenham um controle maior sobre o processo de impressão. As modificações incluem:

- Diálogos de configuração de página: os diálogos padrão e personalizado podem ser exibidos. O usuário pode definir intervalos de páginas, tamanho do papel e o dimensionamento antes de imprimir.
- Visualização da impressão: um modo de visualização pode ser criado para exibir com precisão o tamanho do papel, margens e o a posição do conteúdo na página.
- Impressão restrita: os autores podem restringir as opções de impressão como, por exemplo, o intervalo de páginas que podem ser impressas.
- Opções de qualidade: os autores podem ajustar a qualidade da impressão para um documento e permitir que o usuário selecione a resolução e opções de cor.
- Diversas sessões de impressão: uma instância única PrintJob pode ser utilizada para diversas sessões de impressão. Os aplicativos podem fornecer configurações consistentes a cada momento em que as configurações de página e diálogos de impressão são exibidos.

Alterações no fluxo de trabalho de impressão

O fluxo de trabalho de uma impressão nova consiste nas seguintes etapas:

- `new PrintJob()`: Cria uma instância PrintJob (ou reutiliza uma instância existente). Diversas propriedades PrintJob e métodos como, por exemplo, `selectPaperSize()` estão disponíveis antes da tarefa de impressão iniciar ou durante a impressão.
- `PrintJob.showPageSetupDialog()`: (opcional) Exibe o diálogo de configuração da página sem iniciar uma tarefa de impressão.
- `PrintJob.start()` ou `PrintJob.start2()`: Além do método `start()`, o novo método `start2()` é utilizado para iniciar o processo de spooling de impressão. O método `start2()` permite selecionar quando o diálogo de Impressão e personalizar o diálogo quando ele é exibido.
- `PrintJob.addPage()`: Adicionar conteúdo à tarefa de impressão. Sem modificações no processo existente.
- `PrintJob.send()` ou `PrintJob.terminate()`: Envia as páginas para a impressora selecionada ou encerra o tarefa de impressão sem enviar. As tarefas de impressão são encerradas em resposta a um erro. Se uma instância PrintJob é encerrada, ela ainda pode ser reutilizada. Independentemente da tarefa de impressão ser enviada para a impressora ou encerrada, as configurações atuais de impressão são mantidas quando a instância PrintJob é reutilizada.

Caixa de diálogo Configurar Página

O método `showPageSetupDialog()` exibe o diálogo de Configuração de impressão do sistema, se o ambiente atual tiver suporte. Sempre verifique a propriedade`supportsPageSetupDialog` antes de chamar este método. A seguir, um exemplo simples:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
//check for static property supportsPageSetupDialog of PrintJob class
if (PrintJob.supportsPageSetupDialog) {
    myPrintJob.showPageSetupDialog();
}
```

Opcionalmente, o método pode ser chamado com uma classe PrintUIOptions propriedade para controlar quais opções são exibidas no diálogo de Configurações de impressão. O número mínimo e máximo de páginas que pode ser definido. O exemplo a seguir limita a impressão para as três primeiras páginas:

```
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);
}
```

Alteração das configurações de impressão

As configurações da instância PrintJob podem ser alteradas a qualquer momento após ela ser construída. Isto inclui alterar configurações entre chamadas `addPage()` e após a tarefa de impressão ter sido enviada ou encerrada. Algumas configurações como a propriedade`printer`, aplicam-se à tarefa de impressão toda e não a páginas individuais. Algumas configurações devem ser definidas antes de uma chamada para `start()` ou `start2()`.

O método `selectPaperSize()` pode ser chamado para definir o tamanho de papel padrão nos diálogos de Configurações da página e Configurações de impressão. Ele também pode ser chamado durante uma tarefa de impressão para definir o tamanho do papel para um intervalo de páginas. Ele é chamado por constantes definidas na classe `PaperSize` como nestes exemplo, que seleciona um número 10 de tamanho de envelope:

```
import flash.printing.PrintJob;
import flash.printing.PaperSize;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.selectPaperSize(PaperSize.ENV_10);
```

Utilize a propriedade`printer` para obter ou definir o nome da impressora para o trabalho atual. Por padrão, isto é definido com o nome da impressora padrão. A propriedade `printer` é `null` nenhuma impressora estiver disponível ou o sistema não tiver suporte à impressão. Para alterar a impressora, primeiro obtenha uma lista de impressoras disponíveis utilizando a propriedade `printers`. Essa propriedade é um Vetor cujo os elementos de Sequência de caracteres são os nomes das impressoras disponíveis. Defina a propriedade `printer` para um desses valores de Sequência de caracteres para tornar a impressora ativa. A propriedade `printer` de um trabalho de impressão ativo não pode ser modificada. Tenta mudar após uma chamada com sucesso para `start()` ou `start2()` e antes da tarefa ser enviada ou o terminal falhar. Segue um exemplo de configuração dessa propriedade:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.printer = "HP_LaserJet_1";
myPrintJob.start();
```

A propriedade `copies` obtém o valor do número de cópias definidas no diálogo Imprimir do sistema operacional. As propriedades `firstPage` e `lastPage` obtém o intervalo de páginas. A propriedade `orientation` obtém as configurações de orientação de página. Essas propriedades podem ser definidas para sobrescrever os valores do diálogo Imprimir. O exemplo a seguir define essas propriedades:

```
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;
```

A configuração somente leitura a seguir associada com `PrintJob` fornece informações úteis sobre a configuração atual da impressora:

- `paperArea`: representa em pontos os limites retangulares do papel da impressora.
- `printableArea`: representa em pontos os limites retangulares da área que pode ser impressa.
- `maxPixelsPerInch`: representa em pixels por polegada a resolução física da impressora atual.
- `isColor`: A habilidade da impressora de imprimir em cores (retorna `true` se a impressora pode imprimir em cores).

Consulte "Exemplo de impressão: configuração de página e opções de impressão" na página 966.

Exemplo de impressão: impressão de várias páginas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao imprimir mais de uma página de conteúdo, você pode associar cada página de conteúdo a uma entidade gráfica diferente (neste caso, `sheet1` e `sheet2`) e usar `PrintJob.addPage()` para cada entidade gráfica. O código a seguir ilustra esta técnica:

```
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();
                }
            }
        }
    }
}
```

Exemplo de impressão: escala, corte e resposta

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Em alguns casos, você pode desejar ajustar o tamanho (ou outras propriedades) do objeto de exibição durante a impressão para acomodar diferenças entre a forma como ele é exibido na tela e como é impresso no papel. Quando você ajusta as propriedades de um objeto de exibição antes da impressão (por exemplo, usando as propriedades `scaleX` e `scaleY`), lembre-se de que, se o objeto tiver a escala maior do que o retângulo definido para a área de impressão, ele será cortado. Talvez você também queira redefinir as propriedades depois que as páginas foram impressas.

O código a seguir escala as dimensões do objeto de exibição `txt` (mas não o plano de fundo da caixa verde) e o campo de texto acaba sendo cortado pelas dimensões do retângulo especificado. Após a impressão, o campo de texto volta ao seu tamanho original para exibição na tela. Se o usuário cancelar o trabalho de impressão da caixa de diálogo Imprimir do sistema operacional, o conteúdo no tempo de execução do Flash será alterado para alertar o usuário de que o trabalho foi cancelado.

```
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;
        }
    }
}
```

Exemplo de impressão: configuração de página e opções de impressão

Adobe AIR 2 e posterior

O exemplo a seguir inicializa as configurações de PrintJob para o número de cópias, tamanho de papel (ofício) e orientação da página (paisagem). Ela força a exibição do diálogo de Configurações da página e, em seguida, inicia a tarefa de impressão, exibindo o diálogo Imprimir.

```
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;
        }
    }
}
```

Capítulo 55: Geolocalização

Caso um dispositivo suporte geolocalização, você pode utilizar a API de geolocalização para obter a localização geográfica atual do dispositivo. Caso o dispositivo suporte este recurso, você pode obter informação sobre geolocalização. Estas informações incluem a altitude, direção, velocidade e carimbo de data e hora da última mudança na localização.

A classe Geolocation despacha eventos `update` em resposta ao sensor de local do dispositivo. O evento `update` é um objeto GeolocationEvent.

Mais tópicos da Ajuda

flash.sensors.Geolocation

flash.events.GeolocationEvent

Detectando mudanças de geolocalização

Para utilizar o sensor de geolocalização, instancie um objeto Geolocation e registre para `atualizar` eventos enviados. O evento `update` é um objeto de evento Geolocation. O evento possui oito propriedades:

- `altitude`—Altitude em metros.
- `heading`—A direção do movimento (com respeito ao Norte Verdadeiro) em graus de número inteiro.
- `horizontalAccuracy`—A exatidão horizontal em metros.
- `latitude`—A latitude em graus.
- `longitude`—A longitude em graus.
- `speed`—A velocidade em metros por segundo.
- `timestamp`—O número de milissegundos no momento do evento a partir do início da execução.
- `verticalAccuracy`—A exatidão vertical em metros.

A propriedade `timestamp` é um objeto int. Os outros são objetos Number.

Este é um exemplo básico que exibe dados de geolocalização em um campo de texto:

```
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()
}
```

Para utilizar este exemplo, certifique-se de criar o campo de texto `geoTextField` e adicioná-lo para a lista de exibição antes de utilizar este código.

Você pode ajustar o intervalo de tempo para eventos de geolocalização chamando o método `setRequestedUpdateInterval()` do objeto Geolocation. Este método usa um parâmetro `interval`, que é o intervalo de atualização solicitado em milisegundos:

```
var geo:Geolocation = new Geolocation();
geo.setRequestedUpdateInterval(10000);
```

A hora real entre atualizações de geolocalização pode ser maior ou menor do que este valor. Qualquer mudança no intervalo de atualização afeta todos os ouvintes registrados. Se o método `setRequestedUpdateInterval()` não for chamado, o aplicativo recebe atualizações com base no intervalo padrão do dispositivo.

O usuário pode impedir que um aplicativo acesse dados de geolocalização. Por exemplo, o iPhone consulta o usuário quando um aplicativo tenta obter dados de geolocalização. Em resposta à consulta, o usuário pode negar o acesso do aplicativo aos dados de geolocalização. O objeto Geolocation envia um evento `status` quando o usuário não fornece acesso aos dados de geolocalização. Além disso objeto Geolocation possui uma propriedade `muted` que é definida como `true` quando o sensor de geolocalização não está disponível. O objeto Geolocation envia um evento `status` quando a propriedade `muted` muda. O código a seguir mostra como detectar quando os dados de geolocalização não estão disponíveis:

```
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 = "";
        }
    }
}
```

Nota: *iPhones de primeira geração que não possuíam uma unidade GPS, enviam eventos de* `update` *ocasionalmente. Nestes dispositivos, o objeto Geolocation inicialmente envia um ou dois eventos*`update`*. Em seguida, ele envia eventos* `update` *quando as informações são alteradas consideravelmente.*

Verificando o suporte a geolocalização

Utilize a propriedade `Geolocation.isSupported` para testar o ambiente de tempo de execução em relação a habilidade de utilizar este recurso:

```
if (Geolocation.isSupported)
{
    // Set up geolocation event listeners and code.
}
```

Atualmente, a geolocalização é suportada apenas em aplicativos com base no ActionScript para o iPhone e no Flash Lite 4. Caso `Geolocation.isSupported` for `true` no tempo de execução, então o suporte a geolocalização está disponível.

Alguns modelos de iPhone não tem uma unidade GPS. Estes modelos utilizam outros meios (assim como triangulação de telefone móvel) para obter dados de geolocalização. Para estes modelos, ou para qualquer modelo de iPhone que não tenham GPS desabilitado, um objeto Geolocation pode enviar apenas um ou dois eventos de`atualização` iniciais.

Capítulo 56: Internacionalização de aplicativos

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

O pacote flash.globalization torna mais fácil criar softwares internacionais que se adaptam às convenções de diferentes idiomas e regiões.

Mais tópicos da Ajuda

Pacote flash.globalization

"Localização de aplicativos" na página 991

Charles Bihis: Want to Localize your Flex/AIR Apps? (Deseja Localizar seus Aplicativos Flex/AIR)

Introdução à internacionalização de aplicativos

Em alguns casos, os termos globalização e internacionalização são utilizados alternadamente. No entanto, as definições mais populares desses termos dizem que a globalização refere-se à combinação de processos de negócios e engenharia enquanto a internacionalização refere-se somente à engenharia.

Seguem algumas definições de termos importantes:

Globalização Uma grande variedade de processos de engenharia e negócios necessários para preparar e lançar produtos e atividades da companhia globalmente. A globalização consiste em atividades de engenharia como, por exemplo, internacionalização, localização e culturalização e atividades de negócios como gerenciamento de produto, planejamento financeiro, marketing e trabalho jurídico. Globalization as vezes é abreviado como *G11n* (letra G, mais 11 letras e n). *"Globalização é o que a empresa faz."*

Internacionalização Processo de engenharia para generalizar um produto de forma que ele possa utilizar diversos idiomas, scripts e convenções culturais (incluindo moedas, regras de classificação, formatos de data e números, etc.) sem necessidade de um novo design ou nova compilação. Esse processo pode ser dividido em dois conjuntos de atividades: capacitação e localização. Às vezes a internacionalização é conhecida como *world-readiness* e é abreviada como *I18n*. *"A internacionalização é o que os engenheiros fazem."*

Localização Processo de adaptação de um produto ou serviço para uma cultura e idioma específico e a aparência local desejada. Às vezes a localização é abreviada como *L10n*. *"Localização é o que os tradutores fazem."*

Culturização Processo de engenharia para desenvolvimento ou adaptação de recursos específicos às necessidades de uma cultura. Alguns exemplos incluem os recursos de publicação para o idioma japonês disponíveis no Adobe InDesign e o recursos de suporte a Hanko no Adobe Acrobat.

Alguns termos importantes de internacionalização podem ser definidos da seguinte forma:

Conjunto de caracteres Caracteres utilizados pelo idioma ou por um grupo de idiomas. Um conjunto de caracteres inclui caracteres nacionais e especiais (ex: pontuação e símbolos matemáticos), dígitos numéricos e caracteres controlados por computador.

Ordem alfabética Classificação de texto na ordem adequada para uma localidade específica.

Localidade Valor que representa as convenções culturais e de idioma utilizadas em uma região geográfica, política ou cultural (o que muitas vezes pode indicar um único país). Um identificador de localidade único (ID de localidade) representa este valor. O ID de localidade é utilizado para definir dados de localidade que fornecem suporte específico da localidade. O suporte aplica-se a unidades de medida, à análise e formatação de números, datas e assim por diante.

Pacote de recursos Conjunto armazenado de elementos específicos de localidade criados para uma localidade na qual um aplicativo é utilizado. Um recurso geralmente contém todos os elementos de texto na interface de usuário do aplicativo. Dentro de um pacote, os elementos são traduzidos no idioma apropriado para a localidade especificada. Ele também pode conter outras configurações como o layout ou comportamento da interface de usuário de uma localidade específica. Um pacote de recurso pode conter outros tipos de mídia ou referências a outros tipos de mídia, específicas do local.

Visão geral do pacote flash.globalization

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

O pacote flash.globalization utiliza os recursos de suporte cultural do sistema operacional base. Ele torna mais fácil escrever aplicativos que seguem as convenções culturais de usuários individuais.

As principais classes do pacote incluem:

- A classe Collator que governa a classificação e correspondência das strings.
- A classe CurrencyFormatter que formata números em strings de valores de moeda e analisa os símbolos e valores de moedas de strings de entrada
- A classe DateTimeFormatter que formata os valores de data
- A classe LocaleID para obter informações sobre uma localidade especifica
- A classe NumberFormatter que formata e analisa valores numéricos
- A classe StringTools que trata de conversões de strings sensíveis a localidades

O pacote flash.globalization e localização de recursos

O pacote flash.globalization não manipula localização de recursos. No entanto você pode utilizar a ID de localidade flash.globalization como o valor chave para obter recursos localizados utilizando outras técnicas. Por exemplo, você pode localizar recursos de aplicativo criados com o Flex usando as classes ResourceManager e ResourceBundle. Para mais informações, acesse Localizando aplicativos Flex.

O Adobe AIR 1.1 possui também alguns recursos para ajudar na localização de aplicativos AIR como discutido em "Localização de aplicativos AIR" na página 992.

Uma abordagem geral para internacionalização de um aplicativo

Os seguintes passos descrevem uma aproximação de alto nível para internacionalizar um aplicativo utilizando o pacote flash.globalization:

1. Determinar ou definir a localidade.
2. Criar uma ocorrência de uma classe de serviço (Collator, CurrencyFormatter, DateTimeFormatter, NumberFormatter ou StringTools).
3. Verifique se existem erros e reduções de velocidade utilizando as propriedades lastOperationStatus.

4 Formatar e exibir informações utilizando configurações de localidade especifica.

O próximo passo é carregar e exibir strings e recursos UI (interface do usuário) que são especificas para a localidade. Esta etapa pode incluir tarefas como, por exemplo:

- Utilização de recursos autolayout para redimensionar a UI para acomodar comprimentos de sequência de caracteres
- Selecionar as fontes corretas e os fontes reserva de suporte
- Utilizar o mecanismo de texto FTE para fornecer suporte a outros sistemas de escrita
- Certificar-se de que os editores de método de entrada são tratados corretamente

Verificando erros e reduções de velocidade

As classes de serviço flash.globalization seguem um padrão similar para identificar erros. Elas também compartilham um padrão para retornar de uma localidade solicitada indisponível para uma que possua suporte do sistema operacional.

O exemplo a seguir mostra como verificar a existência de erros e fallbacks ao instanciar classes de serviço. Cada classe de serviço possui uma propriedade lastOperationStatus que indica se a chamada de método mais recente disparou erros ou avisos.

```
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);
    }
}
```

Este exemplo apenas rastreia uma mensagem se uma ID de localidade fallback é utilizada ou se há um erro. Seu aplicativo pode executar lógicas de manipulação de erros adicionais, caso seja necessário. Por exemplo, você pode exibir uma mensagem para o usuário ou forçar o aplicativo a utilizar uma localidade especifica suportada.

Determinando a localidade

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

Uma localidade identifica uma combinação especifica de linguagem e convenções culturais para um país ou região.

Um identificador de localidade pode ser gerenciado de forma segura como uma string. Mas você pode utilizar a classe LocaleID para obter informações adicionais relacionadas à localidade.

Você pode criar um objeto LocaleID da seguinte forma:

```
var locale:LocaleID = new LocaleID("es-MX");
```

Após o objeto LocaleID ser criado, você pode recuperar os dados do ID de localidade. Utilize os métodos `getKeysAndValues()`, `getLanguage()`, `getRegion()`, `getScript()`, `getVariant()` e `isRightToLeft()` e a propriedade `name`.

Os valores obtidos destes métodos e propriedades podem refletir informações adicionais sobre a localidade, que não podem ser extraídas diretamente do identificador de localidade.

Quando um aplicativo cria um serviço de informações sobre localidade, como um formatador de data, a localidade tem que ser especificada. A lista de localidades que possuem suporte pode variar conforme o sistema operacional, portanto, a localidade solicitada pode não estar disponível.

Primeiro o Flash Player tenta combinar o código de linguagem solicitado. Então ele tenta refinar a localidade procurando um sistema de escrita e região correspondentes (script) Por exemplo:

```
var loc:LocaleID = new LocaleID("es");
trace(loc.getLanguage()); // es
trace(loc.getScript()); // Latn
trace(loc.getRegion()); // ES
```

Neste exemplo o construtor `LocaleID()` obteve dados sobre a localidade que corresponde melhor ao código de linguagem "es" para o usuário.

Definindo o ID de localidade

Existem algumas maneiras de definir a localidade atual para um aplicativo, incluindo:

- Codificando um ID de localidade no aplicativo. Esta é uma abordagem comum, mas não suporta a internacionalização do aplicativo.
- Utilizar as preferências de ID de localidade do sistema operacional do usuário, ou do navegador, ou de outras preferências de usuário. Isto geralmente resulta nas melhores configurações de localidade para o usuário, mas nem sempre possui precisão. Existe um risco de que as configurações do sistema operacional não reflitam as preferências em uso do usuário. Por exemplo, o usuário pode estar utilizando um computador compartilhado e ser capaz de mudar as localidades preferidas do sistema operacional.
- Depois de definir o ID de localidade com base nas preferências do usuário, permita ao usuário selecionar em uma lista de localidades suportadas. Geralmente, esta estratégia é a melhor opção caso o seu aplicativo possa suportar mais de uma localidade.

Você pode implementar esta terceira opção da seguinte forma:

1. Obtenha uma lista das localidades ou linguagens preferidas pelo usuário de um perfil de usuário, configurações de navegador, configurações de sistema operacional ou de um cookie. (Seu aplicativo deverá implementar sua esta lógica por si. A biblioteca flash.globalization não suporta leitura direta de tais preferências).
2. Determine quais dessas localidades são suportadas por seu aplicativo e selecione a melhor por padrão. Utilize o método LocaleID.determinePreferredLocales() para encontrar as melhores localidades para um usuário com base em suas localidades preferidas e as localidades suportadas pelo sistema operacional
3. Forneça ao usuário uma maneira de alterar a configuração de localidade padrão no caso da localidade padrão não seja satisfatória.

Limitações de outras classes de localidade e idioma

A classe `fl.lang.Locale` permite que você substitua strings de texto com base em uma localidade, utilizando pacotes de recursos contendo valores de string. No entanto, esta classe não suporta outros recursos de internacionalização como número, moeda ou formatação de data, classificação e comparação, e assim por diante. Além disso esta classe está disponível apenas no Flash Professional.

Você também pode obter a configuração atual do código de linguagem do sistema operacional utilizando a propriedade `flash.system.Capabilities.language`. No entanto, essa propriedade recupera somente códigos de idioma ISO 639-1 de caracteres duplos - e não o ID de localidade completo — e somente possui suporte a um conjunto específico de localidades.

No AIR 1.5, é possível utilizar a propriedade `flash.system.Capabilities.languages`. Essa propriedade fornece um conjunto de idiomas de interface preferidos dos usuário. Portanto, não possui as limitações de `Capabilities.language`.

Formatando números

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

O formato de exibição de valores numéricos variam muito de região para região. Por exemplo, o número 123456.78 é formatado para certas localidades da seguinte maneira:

Localidade	Formato do número
en-US (English, USA)	-123,456.78
de-DE (German, Germany)	-123.456,78
fr-FR (France, French)	-123 456,78
de-CH (German, Switzerland)	-123'456.78
en-IN (English, India)	-1,23,456.78
Várias localidades Árabes	123,456.78-

Existem vários fatores que influenciam formatos de número, incluindo:

- Separadores. O separador decimal é colocado entre as partes inteira e fracionário de um número. Pode ser um ponto, uma vírgula ou outro caractere. O separador de agrupamento ou de milhares pode ser um ponto, uma vírgula, um espaço sem quebra ou outro caractere.
- Padrões de agrupamento. O número de dígitos entre cada separador de agrupamento à esquerda do ponto decimal pode ser dois ou três ou outro valor.
- Indicadores de números negativos. Números negativos podem ser mostrados com um sinal de menos à esquerda ou à direita do número, ou dentro de parênteses para aplicativos financeiros. Por exemplo, 19 negativo pode ser mostrado como -19, 19- ou (19).
- Zeros à esquerda e à direita. Algumas convenções culturais adicionam zeros à esquerda ou à direita em números exibidos. Por exemplo, o valor 0.17 pode ser exibido como .17, 0.17, ou 0.170, entre outras opções.
- Conjunto de caracteres de dígito. Muitas linguagens, incluindo Hindi, Árabe e Japonês utilizam diferentes conjuntos de caracteres de dígitos. O pacote flash.globalization suporta quaisquer conjuntos de caracteres de dígitos que mapeiam para os dígitos 0-9.

A classe NumberFormatter considera todos estes fatores ao formatar valores numéricos.

Utilizando a classe NumberFormatter

A classe NumberFormatter formata valores numéricos (do tipo int, uint, ou Number) de acordo com as convenções de uma localidade especifica.

O exemplo a seguir mostra a maneira mais simples de formatar um número utilizando as propriedades de formatação padrão fornecidas pelo sistema operacional do usuário:

```
var nf:NumberFormatter = new NumberFormatter(LocaleID.DEFAULT);
trace(nf.formatNumber(-123456.789))
```

O resultado pode variar com base nas configurações de localidade e das preferências do usuário. Por exemplo, se a localidade do usuário é fr-FR o valor formatado deve ser:

-123.456,789

Caso você queira apenas formatar um número para uma localidade especifica, independente das configurações de usuário, defina o nome da localidade especificamente. Por exemplo:

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
trace(nf.formatNumber(-123456.789));
```

Nesse caso, o resultado é:

-123'456.789

O método formatNumber() obtém um Number como parâmetro. A classe NumberFormatter também tem um método formatInt() que obtém um int (inteiro) como entrada e um método formatUint() que obtém um uint.

Você pode controlar explicitamente a lógica de formatação definindo as propriedades da classe NumberFormatter, como mostra este exemplo:

```
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)
```

Este exemplo cria primeiro um objeto NumberFormatter e, em seguida:

- define o formato de número negativo para utilizar parênteses (como em aplicativos financeiros);
- define o número de dígitos depois do separador decimal para 5;
- especifica que os zeros à direita devem ser utilizados para garantir cinco casas decimais,
- define o separador decimal como vírgula;
- informa ao formatador para não utilizar qualquer separador de agrupamento.

Nota: *Caso alguma alteração seja feita nessas propriedades, o formato de número resultante não corresponderá ao formato de preferência para a localidade especificada. Utilize algumas dessas propriedades somente quando a detecção de localidade não for importante, quando é preciso obter um controle detalhado sobre um único aspecto do formato como, por exemplo, o número de zeros à direita ou quando o usuário solicitar a alteração diretamente, pelo Painel de Controle do Windows.*

Analisando strings que contém valores numéricos

A classe NumberFormatter pode também extrair valores numéricos de strings que se ajustem às necessidades de formatação especificas de localidade. O método NumberFormatter.parseNumber() extrai um valor numérico único de uma string. Por exemplo:

```
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
```

O método parseNumber() manipula strings que contém caracteres de formatação de dígitos e números como sinais negativos e separadores. Se a cadeira de caracteres contiver outros caracteres, um código de erro é definido:

```
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
```

Para extrair números de strings que contém caracteres de alfabeto adicionais, utilize o método NumberFormatter.parse:

```
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
```

O método parse() retorna um objeto NumberParseResult que contém o valor numérico analisado em sua propriedade de valor. A propriedade startIndex indica o índice do primeiro caractere numérico encontrado. Você pode utilizar também as propriedades startIndex e endIndex para extrair as porções da string que venham antes e depois dos dígitos.

Formatando valores de moeda

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

Os formatos de exibição de valores de moeda variam tanto quanto os formatos de números Por exemplo, o valor em dólar $123456.78 é formatado para certas localidades da seguinte maneira:

Localidade	Formato do número
en-US (English, USA)	$123,456.78
de-DE (German, Germany)	123.456,78 $
en-IN (English, India)	$ 1,23,456.78

A formatação de moeda envolve todos os mesmos fatores da formatação de números, mais os adicionais:

- Código ISO de moeda. As três letras de código de moeda ISO 4217 para a localidade sendo utilizada, assim como USD ou EUR.
- Símbolo de moeda. O símbolo de moeda ou sequência de caracteres da localidade sendo utilizada, como $ ou €.

- Formato de moeda negativo. Define o local do símbolo de moeda e o símbolo negativo ou parênteses em relação à porção numérica do valor da moeda.
- Formato de moeda positivo. Define o local do símbolo de moeda quanto à porção numérica do valor da moeda.

Utilizando a classe CurrencyFormatter

A classe CurrencyFormatter formata valores numéricos em strings que contém strings de moeda e números formatados, de acordo com as convenções de uma localidade especifica.

Quando você instancia um novo objeto CurrencyFormatter, ele define a moeda para o padrão para a localidade fornecida.

O exemplo a seguir mostra que um objeto CurrencyFormatter criado com o uso da localidade alemã assume que as quantidades de moeda estão em Euros:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "de-DE" );
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

Na maioria dos casos você não deve contar com o padrão de moeda para uma localidade. Caso não existe suporte para a localidade padrão do usuário, a classe CurrencyFormatter define uma localidade fallback. A localidade fallback pode ter uma moeda padrão diferente. Além disso, você desejará que os formatos de moeda pareçam corretas para seu usuário, mesmo se as quantidades não estejam na moeda local do usuário. Por exemplo, um usuário canadense pode querer ver os preços de uma empresa alemã em euros, mas formatados no estilo canadense.

O método CurrencyFormatter.setCurrency() especifica a cadeia de caracteres de moeda exata e o símbolo da moeda a ser utilizado.

O exemplo a seguir mostra quantidades de moeda em Euros para usuários na parte francesa do Canadá:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "fr-CA" );
cf.setCurrency("EUR", "€");
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

O método setCurrency() pode também ser utilizada para reduzir a confusão ao definir símbolos de moeda sem ambiguidade. Por exemplo:

```
cf.setCurrency("USD","US$");
```

O método format() exibe um código de moeda IDO 4217 de três caracteres ao invés do símbolo de moeda. Os códigos ISO 4217 não possuem ambiguidade e não requerem localização. No entanto muitos usuários preferem ver símbolos de moeda ao invés de códigos ISO.

A classe CurremcyFormatter pode ajudá-lo a decidir se uma string de moeda formatada deve utilizar um símbolo de moeda, como um símbolo de dólar ou de euro, ou uma string de moeda ISO de três caracteres, como USD ou EUR. Por exemplo, um valor em dólares Canadenses pode ser exibido como $200 para um usuário no Canadá. No entanto, para um usuário nos EUA, ele pode ser exibido como CAD 200. Utilize o método formattingWithCurrencySymbolIsSafe() para determinar se o símbolo de moeda de quantidade seria ambíguo ou incorreto para em relação às configurações de localidade do usuário.

O exemplo a seguir formata um valor em euros em um formato par a localidade en-US. Dependendo da localidade do usuário, a string de saída usará o código de moeda ISO ou o símbolo de moeda.

```
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
}
```

Analisando strings que contém valores numéricos de moeda

A classe CurrencyFormatter pode também extrair uma string de quantidade de moeda de uma string de entrada que corresponda às necessidades de formatação especificas de localidade. O método CurrencyFormatter.parse() armazena a quantidade analisada e a string de moeda em um objeto CurrencyParseResult, como mostrado em:

```
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
```

A porção de string de moeda da string de entrada pode conter um símbolo de moeda, um código ISO de moeda, e caracteres adicionais de texto. As posições da cadeia de caracteres de moeda, o indicador de número negativo e o valor numérico, devem corresponder aos formatos especificados pelas propriedades negativeCurrencyFormat e positiveCurrencyFormat. Por exemplo:

```
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 $
```

Neste exemplo, a string de entrada tem uma string de moeda seguida por um sinal de menos e um número. No entanto o valor padrão de negativeCurrencyFormat para a localidade en-US especifica que o indicador negativo deve vir primeiro. Como resultado, o método parse() gera um erro e o valor analisado é NaN.

Em seguida ele define o negativeCurrencyFormat para 2, o qual especifica que a string de moeda vem primeiro, o método parse() terá êxito.

Formatando data e hora

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

O formato de exibição de valores de data e tempo também variam muito de uma região para outra. Por exemplo, aqui está como o segundo dia de janeiro, de 1962 às 1:01 PM é exibido em um formato curto para certas localidades:

Localidade	Formato de data e hora
en-US (English, USA)	2/1/62 1:01pm
fr-FR (France, French)	2/1/62 13:01
ja-JP (Japan, Japanese)	1962/2/1 13:01

Utilizando a classe DateFormatter

A classe DateTimeFormatter formata valores Date em strings de data e hora de acordo com as convenções de uma localidade especifica.

A formatação é orientada por uma string de padrão que contém sequências de letras que serão substituídas por valores de data ou tempo. Por exemplo, no padrão "aaaa/MM", os caracteres o "aaaa" será substituído por um ano de quatro dígitos, seguido pelo caractere "/" e um mês de dois dígitos.

A string de padrão pode ser definida explicitamente utilizando o método setDateTimePattern(). No entanto, é melhor permitir que o padrão seja definido automaticamente, de acordo com a localidade do usuário e as preferências do sistema operacional. Esta prática ajuda a garantir que o resultado seja apropriado de forma cultural.

O DateTimeFormatter pode representar datas e horários em três estilos padrão (LONG, MEDIUM e SHORT) além de poder utilizar um padrão CUSTOM. Um padrão pode ser utilizado para data, e o secundo para a hora. Os padrões reais utilizados para cada estilo podem variar um pouco entre sistemas operacionais.

É possível especificar estilos ao criar um objeto DateTimeFormatter. Caso os parâmetros de estilo não estejam especificados, então eles serão definidos para DateTimeStyle.LONG por padrão. Você pode alterar os estilos posteriormente utilizando o método setDateTimeStyles(), como mostrado no seguinte exemplo:

```
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
```

Localizando nomes de meses e de dias

Muitos aplicativos utilizam listagens de nomes de meses e os nomes dos dias da semana em exibições de calendário ou listas pull-down.

Você pode obter uma lista localizada dos nomes dos meses utilizando o método DateTimeFormatter.getMonthNames(). Dependendo do sistema operacional, formas completas e abreviadas podem estar disponíveis. Passe o valor DateTimeNameStyle.FULL para obter nomes completos de meses. Passe os valores DateTimeNameStyle.LONG_ABBREVIATION ou DateTimeNameStyle.SHORT_ABBREVIATION para obter uma versão menor.

Em alguns idiomas, o nome do mês é mudado (em sua forma genitiva) quando colocado ao lado de um valor de dia em um formato de data. Caso planeje utilizar os nomes de meses separados, deverá passar o valor DateTimeNameContext.STANDALONE para o método getMonthNames(). No entanto, para utilizar nomes de mês em datas formatadas, passe o valor 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.
```

O método DateTimeFormatter.getWeekdayNames() fornece uma lista localizada dos nomes dos dias da semana. O método getWeekdayNames() aceita os mesmos parâmetros de contexto e nameStyle que o método getMonthNames() aceita.

```
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.
```

Além disso, o método getFirstWeekday() retorna o valor de índice do dia que tradicionalmente marca o começo da semana na localidade selecionada.

Classificando e comparando strings

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

Comparação é o processo de arrumar coisas na ordem apropriada. As regras de comparação variam bastante por localidade. As regras também são diferentes caso esteja classificando uma lista ou correspondendo itens similares como, por exemplo, um algoritmo de busca de texto.

Quando estiver classificando, pequenas diferenças como letras maiúsculas e minúsculas ou marcas diacríticas como acentos são geralmente significativos. Por exemplo, a letra ö (o com trema) é considerado na maioria dos casos como equivalente à letra simples o em francês ou inglês. No entanto, a mesma letra segue a letra z em sueco. Além disso, em francês e algumas outras linguagens, a última diferença de acento em uma palavra determina sua ordem em uma lista classificada.

Ao fazer uma busca, você geralmente vai querer ignorar diferenças entre maiúsculas e minúsculas ou diacríticos, para aumentar a chance de encontrar resultados relevantes. Por exemplo, uma busca pelos caracteres "cote" em um documento em francês deve provavelmente retornar resultados para “cote”, “côte” e “coté”.

Utilizando a classe Collator

Os principais métodos da classe Collator são o compare() que é utilizado primariamente para classificar, e o equals() utilizado para comparar valores.

O exemplo a seguir mostra os diferentes comportamentos dos métodos compare() e equals().

```
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
}
```

O exemplo primeiro cria um objeto Collator no modo SORTING para a localidade French-France. Ele então classifica duas palavras que diferem pelas marcas diacríticas apenas. Isto mostra que a comparação SORTING diferencia entre caracteres com e sem acento.

A classificação é feita passando uma referência para o método sort() do objeto Collator como um parâmetro para o método Array.sort(). Esta é uma das maneiras mais eficientes de utilizar um objeto Collator para controlar a ordem de classificação.

O exemplo cria então um objeto Collator no modo MATCHING. Quando este objeto Collator compara as duas palavras, ele trata as duas como iguais. Isto mostra que a comparação MATCHING valoriza caracteres acentuados ou não da mesma forma.

Personalizando o comportamento da classe Collator

Por padrão, a classe Collator utiliza regras de comparação de string obtidas do sistema operacional com base na localidade e nas preferências do sistema do usuário. Você pode personalizar o comportamento dos métodos compare() e equals() definindo várias propriedades explicitamente. As tabela a seguir lista as propriedades e o efeito que elas tem sobre as comparações:

Propriedade Collator	**Efeito**
numericComparison	Controla se os caracteres de dígito são tratados como números ou texto.
ignoreCase	Controle se as diferenciação entre maiúsculas e minúsculas será ignorada.
ignoreCharacterWidth	Controla se as formas de largura total e parcial de alguns caracteres chineses e japoneses são avaliados como iguais.
ignoreDiacritics	Controla se as strings que utilizam as mesma base de caracteres exceto acentos ou outras marcas diacríticas são avaliadas como iguais.
ignoreKanaType	Controla se as cadeias de caracteres que se diferem apenas pelo tipo de caractere kana em uso serão tratadas como iguais.
ignoreSymbols	Controla se caracteres de símbolo como espaços, símbolos de moeda, símbolos matemáticos e outros são ignorados.

O código a seguir mostra que definir a propriedade ignoreDiacritics para verdadeiro muda a ordem de classificação de uma lista de palavras em francês:

```
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
```

Conversão de caracteres maiúsculos e minúsculos

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

Linguagens diferem também em suas regras para converter letras entre maiúsculas (majiscules) e minúsculas (minuscules).

Por exemplo, a maioria dos idiomas que utilizam o alfabeto Latino a forma da letra minúscula "I" é "i". No entanto em algumas linguagens (como o turco e o azeri) existem uma letra “ı” sem ponto. Como resultado disso, nesses idiomas, um “ı” minúsculo sem acento é transformado em um ”I” maiúsculo. Um “i” minúsculo é transformado em um “İ” maiúsculo sem o ponto.

A classe StringTools fornece métodos que utilizam regras especificas de linguagem para executar tais transformações.

Utilizando a classe StringTools

A classe StringTools fornece dois métodos para executar transformações de maiúsculas e minúsculas: toLowerCase() e toUpperCase(). O objeto StringTools é criado ao chamar o construtor como um ID de localidade. A classe StringTools irá obter as regras de conversão de maiúsculas e minúsculas para a localidade (ou localidade fallback) do sistema operacional. Não é possível fazer mais personalizações no algoritmo de conversão de maiúsculas e minúsculas.

O exemplo a seguir utiliza os métodos toUpperCase() e toLowerCase() para transformar uma frase em alemão que inclui a letra “ß” (sharp S).

```
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
```

O método toUpperCase() transforma a letra minúscula “ß” na letra maiúscula “SS”. Essa transformação funciona somente em uma direção. Quando as letras "SS" são transformadas em minúsculas, o resultado é "ss" e não “ß”.

Exemplo: Internacionalização de um aplicativo de acionamento de ações

Flash Player 10.1 e posterior, Adobe AIR 2.0 e posterior

O aplicativo Acionador de ações globais obtém e exibe dados fictícios sobre ações em três mercados de ações diferentes: dos Estados Unidos, Japão e Europa. Ele formata dos dados de acordo às convenções de várias localidades.

Este exemplo ilustra os seguintes recursos do pacote flash.globalization:

- Formatação de número sensível a localidade
- Formatação de moeda sensível a localidade
- Definindo código ISO e símbolos de moeda
- Formatação de data sensível a localidade
- Obtendo e exibindo abreviações apropriadas de nome de meses

Para obter os arquivos de aplicativo desse exemplo, consulte www.adobe.com/go/learn_programmingAS3samples_flash_br. Os arquivos do aplicativo Acionador de ações globais podem ser localizados no diretório Samples/GlobalStockTicker. O aplicativo consiste nos seguintes arquivos:

Arquivo	Descrição
GlobalStockTicker.mxml ou GlobalStockTicker.fla	A interface do usuário do aplicativo para Flex (MXML) ou Flash (FLA).
styles.css	Estilos da interface de usuário do aplicativo (somente no Flex).
com/example/programmingas3/stockticker/flex/FinGraph.mxml	Um componente MXML que exibe um gráfico de dados de simulados de ações, apenas para o Flex.
com/example/programmingas3/stockticker/flash/GlobalStockTicker.as	Classe Document que contém a lógica da interface de usuário do aplicativo (somente no Flash).
comp/example/programmingas3/stockticker/flash/RightAlignedColumn.as	Um processador de célula personalizado para o componente Flash DataGrid (apenas no Flash).
com/example/programmingas3/stockticker/FinancialGraph.as	Uma classe ActionScript que desenha um gráfico de dados simulados de ações.
com/example/programmingas3/stockticker/Localizer.as	Uma classe ActionScript que gerencia a localidade e a moeda e manipula a formatação localizada de números, quantidades de moeda e datas.
com/example/programmingas3/stockticker/StockDataModel.as	Uma classe ActionScript que armazena todos os dados de amostra para o exemplo do Acionador de ações globais.

Entendendo a interface do usuário e amostra de dados

Os elementos da interface do usuário principal do aplicativo são:

- uma caixa combinada para selecionar uma Localidade
- uma caixa combinada para selecionar um Mercado
- um DataGrid que exibe dados para seis empresas em cada mercado
- um gráfico que mostra dados de histórico simulados para as ações da empresa selecionada

O aplicativo armazena todos os seus dados de amostra sobre localidades, mercados e ações de empresas na classe StockDataModel. Um aplicativo real deve obter dados de um servidor e, em seguida, armazená-los em uma classe como a StockDataModel. Neste exemplo, todos os dados estão codificados na classe StockDataModel.

Nota: *Os dados exibidos no gráfico financeiro não correspondem necessariamente com os dados no controle do DataGrid. O gráfico é redesenhado aleatoriamente cada vez que uma empresa diferente é selecionada. Para propósitos ilustrativos apenas.*

Definindo a localidade

Depois de algum trabalho inicial de configuração, o aplicativo chama o método Localizer.setLocale() para criar objetos formatadores para a localidade padrão. O método setLocale() também é chamado a cada vez que o usuário seleciona um novo valor da caixa combinada de Localidade.

```
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);
    }
}
```

Primeiro o método setLocale() cria um objeto LocaleID. Isto faz com que seja mais fácil obter detalhes sobre a localidade real posteriormente, se necessário.

Em seguida ele cria novos objetos NumberFormatter, CurrencyFormatter e DateTimeFormatte para a localidade. Após criar cada objeto formatador, o método traceError() é chamado. Este método exibe erros e mensagens de avisos no consolte, caso exista um problema na localidade solicitada. (Um aplicativo real deve executar ações com base nestes erros, em vez de somente rastreá-los).

Depois de criar o objeto CurrencyFormatter, o método setLocale() define o código ISO de moeda do formatador, símbolo de moeda e propriedades fractionalDigits para valores préviamente determinados. (Estes valores são definidos cada vez que o usuário seleciona um novo mercado da caixa combinada Mercados).

Depois de criar o objeto DateTimeFormatter, o método setLocale() também obtém uma matriz de abreviações de nomes de meses localizados.

Formatando os dados

Os dados de ações formatados são apresentados em um controle DataGrid. Cada coluna DataGrid chama uma função de rótulo que formata o valor da coluna utilizando o objeto formatador apropriado.

Na versão do Flash, por exemplo, o seguinte código define as colunas DataGrid:

```
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;
```

A versão do Flex do exemplo declara seu DataGrid no MXML. Isto também define funções de rótulo semelhantes para cada coluna.

As propriedades labelFunction se referem às seguintes funções, as quais chamam métodos de formatação da classe Localizer:

```
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);
}
```

Os métodos Localizer então definem e chamam os formatadores apropriados:

```
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);
}
|
```

Capítulo 57: Localização de aplicativos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Localização é o processo de inclusão de recursos para oferecer suporte a vários códigos de idiomas. Código de idiomas é a combinação de um idioma e um código de um país. Por exemplo, en_US se refere ao idioma inglês falado nos Estados Unidos e fr_FR se refere ao idioma francês falado na França. Para localizar um aplicativo para esses códigos de idiomas, você fornece dois conjuntos de recursos: um para o código de idioma en_US e um para o código de idiomas fr_FR.

Os códigos de idiomas podem compartilhar idiomas. Por exemplo, en_US e en_GB (Grã-Bretanha) são códigos de idiomas diferentes. Nesse caso, os dois códigos de idiomas usam o idioma inglês, mas o código do país indica que são códigos de idiomas diferentes e podem, portanto, usar recursos diferentes. Por exemplo, um aplicativo no código de idioma sen_US pode escrever a palavra "color", ao passo que ela seria "colour" no código de idiomas en_GB. Além disso, unidades monetárias seriam representadas em dólares ou em libras, dependendo do código de idiomas, e o formato de data e hora também pode ser diferente.

Você também podem fornecer um conjunto de recursos para um idioma sem especificar o código do país. Por exemplo, você pode fornecer recursos en para o idioma Inglês e fornecer recursos adicionais do código de idiomas en_US, específicos para inglês dos EUA.

A localização vai além de só traduzir as sequências usadas no aplicativo. Ela também pode incluir qualquer tipo de recurso, como arquivos de áudio, imagens e vídeos.

Seleção de código de idiomas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para determinar qual local seu conteúdo ou aplicativo usa, você pode usar um dos seguintes métodos:

- Pacote flash.globalization — Utilize as classes sensíveis a localidades no pacote flash.globalization para obter a localidade padrão para o usuário com base no sistema operacional e nas preferências do usuário. Esta é a aproximação preferida para aplicativos que serão executados no Flash Player 10.1 ou posterior ou AIR 2.0 ou tempos de execução posteriores. Consulte “Determinando a localidade” na página 975 para mais informações.
- Prompt de usuário: você pode iniciar o aplicativo em algum código de idiomas padrão e, então, solicitar ao usuário que selecione o código de idiomas preferido.
- `(Apenas no AIR)Capabilities.languages`: a propriedade `Capabilities.languages` lista uma matriz de idiomas disponíveis nos idiomas de preferência do usuário, conforme definido pelo sistema operacional. As sequências contêm tags de idioma (e informações de script e região, quando aplicável) definidas por RFC4646 (http://www.ietf.org/rfc/rfc4646.txt). As sequências usam hífens como delimitadores (por exemplo, `"en-US"` ou `"ja-JP"`). A primeira entrada na matriz retornada tem a mesma ID de idioma principal da propriedade language. Por exemplo, se `languages[0]` for definido como `"en-US"`, a propriedade `language` será definida como `"en"`. Entretanto, se a propriedade language for definida como `"xu"` (especificando um idioma desconhecido), o primeiro elemento na matriz `languages` é diferente.

- `Capabilities.language`: a propriedade `Capabilities.language` oferece o código de idioma da interface do usuário do sistema operacional. No entanto, essa propriedade está limitada a 20 idiomas conhecidos. E, nos sistemas em inglês, essa propriedade retorna somente o código do idioma, não o código do país. Por esses motivos, é melhor usar o primeiro elemento na matriz `Capabilities.languages`.

Localização de conteúdo Flex

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Adobe Flex inclui uma estrutura para localização de conteúdo Flex. Essa estrutura inclui as classes Locale, ResourceBundle e ResourceManagerImpl, bem como as interfaces IResourceBundle e IResourceManagerImpl.

Uma biblioteca de localização do Flash contendo classes de utilitário para classificar localidades de aplicativos está disponível no Google Code (http://code.google.com/p/as3localelib/).

Mais tópicos da Ajuda

http://code.google.com/p/as3localelib/

Localização de conteúdo Flash

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Adobe Flash Professional inclui a classe Locale nos componentes do ActionScript 3.0. A classe Locale permite controlar como o arquivo SWF exibe o texto multilíngue. O painel Strings do Flash permite usar IDs de string em vez de literais de string nos campos de texto dinâmicos. Essa facilidade permite criar um arquivo SWF que exibe o texto carregado de um arquivo XML específico do idioma. Para obter mais informações sobre como utilizar a classe Locale, consulte Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Localização de aplicativos AIR

Adobe AIR 1.0 e posterior

O AIR SDK oferece uma estrutura de localização HTML (contida no arquivo AIRLocalizer.js). Este framework inclui APIs para auxiliar no trabalho com diversas localidades em um aplicativo com base em HTML. Uma biblioteca ActionScript para classificação de localidades está disponível em http://code.google.com/p/as3localelib/.

Mais tópicos da Ajuda

http://code.google.com/p/as3localelib/

Localização de datas, horas e moedas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A maneira como os aplicativos apresentam dados, horas e moedas varia consideravelmente em cada código de idiomas. Por exemplo, o padrão dos EUA para representação de datas é mês/dia/ano, enquanto o padrão europeu para representação de datas é dia/mês/ano.

Você pode gravar um código para formatar datas, horas e moedas. Por exemplo, o código a seguir converte o objeto Date no formato mês/dia/ano ou no formato dia/mês/ano. Se a variável `locale` (representando o código de idiomas) for definida como `"en_US"`, a função retornará o formato mês/dia/ano. O exemplo converte o objeto Date no formato dia/mês/ano de todos os outros códigos de idiomas:

```
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

A estrutura Flex inclui controles para formatar datas, horas e moedas. Esses controles incluem os controles DateFormatter e CurrencyFormatter.

- mx:DateFormatter
- mx:CurrencyFormatter

Capítulo 58: Sobre o ambiente HTML

Adobe AIR 1.0 e posterior

O Adobe® AIR® usa o WebKit *(www.webkit.org)*, também usado pelo navegador Safari para analisar gerar layout e renderizar o conteúdo HTML e JavaScript. Usar as APIs do AIR em conteúdo HTML é opcional. Você pode programar no conteúdo de um objeto HTMLLoader ou janela HTML inteiramente com HTML e JavaScript. A maioria dos aplicativos e páginas HTML existentes deve ser executada com algumas alterações (supondo que eles usam recursos HTML, CSS, DOM e JavaScript compatíveis com WebKit).

Importante: as novas versões do tempo de execução do Adobe AIR podem incluir versões atualizadas do WebKit. A atualização do WebKit em uma nova versão do AIR *pode* resultar em mudanças inesperadas em um aplicativo do AIR implantado. Essas alterações podem afetar o comportamento ou a aparência do conteúdo HTML em um aplicativo. Por exemplo, os aprimoramentos ou as correções na renderização do Webkit podem alterar o layout dos elementos na interface de usuário de um aplicativo. Por essa razão, é altamente recomendado que você forneça um mecanismo de atualização no seu aplicativo. Caso necessite atualizar seu aplicativo devido a uma alteração na versão do Webkit que vem com o AIR, o mecanismo de atualização do AIR pode solicitar que o usuário instale a nova versão do aplicativo.

A seguinte tabela lista a versão do navegador web Safari que usa a versão de WebKit equivalente à usada no AIR:

Versão do AIR	Versão do Safari
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

Você sempre pode determinar a versão instalada do WebKit examinando a sequência de caracteres de agente do usuário retornado por um objeto HTMLLoader:

```
var htmlLoader:HTMLLoader = new HTMLLoader();
trace( htmlLoader.userAgent );
```

Lembre-se que a versão do WebKit utilizado no AIR não é idêntica a versão de código aberto. Alguns recursos não são suportados no AIR e a versão do AIR pode conter correções de segurança e de falhas que ainda não estão disponíveis na versão correspondente do WebKit. "Recursos do WebKit não suportados no AIR" na página 1010.

Como aplicativos AIR são executados diretamente na área de trabalho, com acesso completo ao sistema de arquivos, o modelo de segurança para conteúdo HTML é mais rigoroso que o modelo de segurança de um navegador da Web típico. No AIR, apenas o conteúdo carregado do diretório de instalação do aplicativo é colocado na *caixa de proteção do aplicativo*. A caixa de proteção do aplicativo tem o nível mais alto de privilégio e permite acesso às APIs do AIR. O AIR coloca outro conteúdo em caixas de proteção isoladas com base na origem do conteúdo. Arquivos carregados do sistema de arquivos entram em uma caixa de proteção local. Arquivos carregados da rede usando os protocolos http: ou https: entram em uma caixa de proteção baseada no domínio do servidor remoto. O conteúdo nessas caixas de proteção que não são de aplicativo não pode acessar nenhuma API do AIR e é executado da mesma forma que em um navegador da Web típico.

O conteúdo HTML no AIR não exibe conteúdo SWF ou PDF se as configurações de alfa, dimensionamento ou transparência forem aplicadas. Para obter informações, consulte "Considerações ao carregar conteúdo SWF ou PDF em uma página HTML" na página 1040 e "Transparência de janelas" na página 923.

Mais tópicos da Ajuda

Referência DOM do Webkit

Referência HTML do Safari

Referência CSS do Safari

www.webkit.org

Visão geral do ambiente HTML

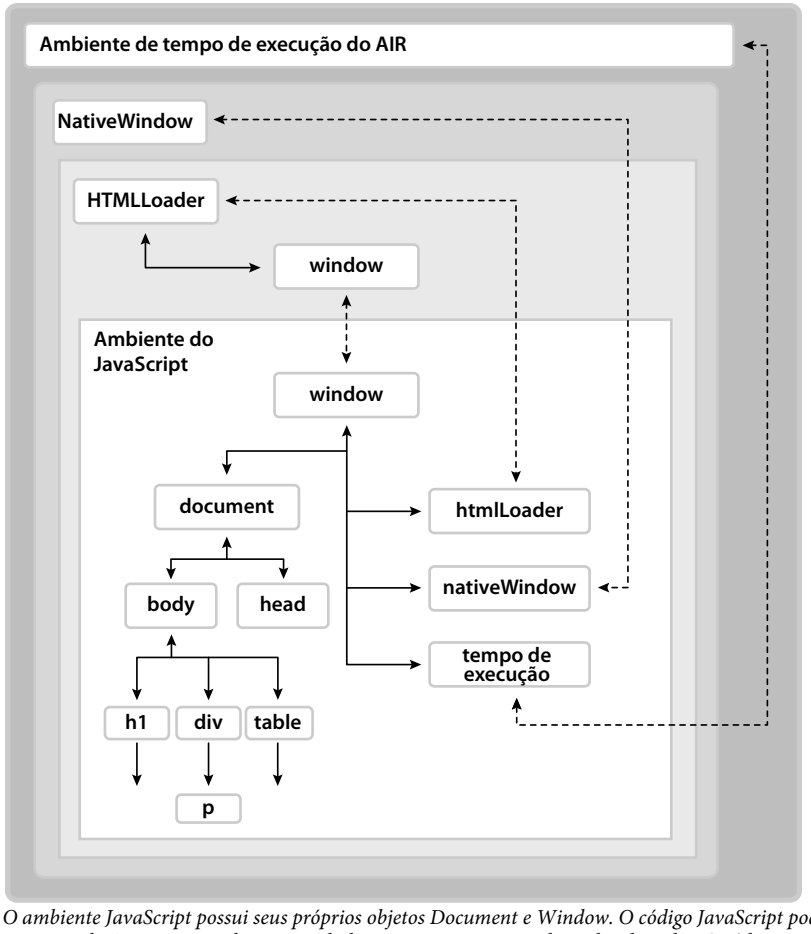
Adobe AIR 1.0 e posterior

O Adobe AIR fornece um ambiente JavaScript completo do tipo de navegador com um processador de HTML, um modelo de objeto de documento e um intérprete do JavaScript. O ambiente JavaScript é representado pela classe HTMLLoader do AIR. Em janelas HTML, um objeto HTMLLoader contém todo o conteúdo HTML e está, por sua vez, contido em um objeto NativeWindow. Em conteúdo SWF, a classe HTMLLoader, que estende a classe Sprite, pode ser adicionada à lista de exibição de um estágio como qualquer outro objeto de exibição. As propriedades da classe do Adobe® ActionScript® 3.0 são descritas em "Script no contêiner HTML do AIR" na página 1038 e na *Referência do ActionScript 3.0 para a Adobe Flash Platform*. Na estrutura do Flex, a classe HTMLLoader do AIR é delimitada em um componente mx:HTML. O componente mx:HTML estende a classe UIComponent, para que possa ser usado diretamente com outros contêineres do Flex. O ambiente JavaScript no componente mx:HTML é, de outro modo, idêntico.

Sobre o ambiente JavaScript e seu relacionamento com o host AIR

Adobe AIR 1.0 e posterior

O diagrama a seguir ilustra o relacionamento entre o ambiente JavaScript e o ambiente de tempo de execução do AIR. Embora apenas uma única janela nativa seja exibida, um aplicativo do AIR pode conter várias janelas. (E uma única janela pode conter vários objetos HTMLLoader.)



O ambiente JavaScript possui seus próprios objetos Document e Window. O código JavaScript pode interagir com o ambiente de tempo de execução do AIR por meio das propriedades runtime, nativeWindow e htmlLoader. O código ActionScript pode interagir com o ambiente JavaScript pela propriedade window de um objeto HTMLLoader, que é uma referência ao objeto Window do JavaScript. Além disso, os objetos ActionScript e JavaScript podem ouvir eventos despachados por objetos AIR e JavaScript.

A propriedade `runtime` fornece acesso a classes API do AIR, permitindo que você crie novos objetos do AIR, bem como membros de classe de acesso (também chamados estáticos). Para acessar uma API do AIR, você adiciona o nome da classe, com pacote, à propriedade `runtime`. Por exemplo, para criar um objeto File, você usaria a instrução:

```
var file = new window.runtime.filesystem.File();
```

Nota: *O SDK do AIR fornece um arquivo JavaScript, `AIRAliases.js`, que define aliases mais convenientes para as classes do AIR usadas mais comumente. Quando você importa esse arquivo, pode usar a forma mais curta air.Class em vez de window.runtime.package.Class. Por exemplo, você poderia criar o objeto File com `new air.File()`.*

O objeto NativeWindow fornece propriedades para controlar a janela da área de trabalho. De uma página HTML, você pode acessar o objeto NativeWindow contido com a propriedade `window.nativeWindow`.

O objeto HTMLLoader fornece propriedades, métodos e eventos para controlar como o conteúdo é carregado e processado. De uma página HTML, você pode acessar o objeto HTMLLoader pai com a propriedade `window.htmlLoader`.

Importante: *Apenas páginas instaladas como parte de um aplicativo possuem as propriedades* `htmlLoader`, `nativeWindow` *ou* `runtime` *e apenas quando carregadas como o documento de nível superior. Essas propriedades não são adicionadas quando um documento é carregado em um frame ou iframe. (Um documento filho pode acessar essas propriedades no documento pai desde que ele esteja na mesma caixa de proteção de segurança. Por exemplo, um documento carregado em um frame poderia acessar a propriedade* `runtime` *de seu pai com* `parent.runtime`*.)*

Sobre a segurança

Adobe AIR 1.0 e posterior

O AIR executa todos os códigos em uma caixa de proteção de segurança baseada no domínio de origem. O conteúdo do aplicativo, limitado ao conteúdo carregado do diretório de instalação do aplicativo, é colocado na caixa de proteção do *aplicativo*. O acesso ao ambiente de tempo de execução e às APIs do AIR está disponível apenas para HTML e JavaScript em execução nessa caixa de proteção. Ao mesmo tempo, a maior parte da execução e avaliação dinâmica de JavaScript é bloqueada na caixa de proteção do aplicativo após todos os manipuladores da página de evento `load` terem sido retornados.

Você pode mapear uma página de aplicativo em uma caixa de proteção que não seja de aplicativo carregando a página em um frame ou iframe e definindo os atributos `sandboxRoot` e `documentRoot` específicos do AIR do frame. Definindo o valor `sandboxRoot` para um domínio remoto real, você pode habilitar o conteúdo da caixa de proteção para cruzar conteúdo de scripts nesse domínio. Mapear páginas dessa maneira pode ser útil ao carregar e fazer o script de conteúdo remoto, como em um aplicativo *mash-up*.

Outra maneira de permitir que o conteúdo de aplicativo e que não é de aplicativo cruze scripts entre si, e a única maneira de fornecer acesso a conteúdo de não aplicativo às APIs do AIR, é criar uma *ponte de caixa de proteção*. Uma ponte *pai para filho* permite conteúdo em um frame filho, iframe ou window para acessar métodos designados e propriedades definidas na caixa de proteção do aplicativo. Por outro lado, uma ponte *filho para pai* permite que conteúdo de aplicativo acesse métodos e propriedades designadas definidas na caixa de proteção do filho. As pontes de caixa de proteção são estabelecidas pela definição das propriedades `parentSandboxBridge` e `childSandboxBridge` do objeto window. Para obter mais informações, consulte "Segurança HTML no Adobe AIR" na página 1121 e "Elementos HTML frame e iframe" na página 1005.

Sobre plug-ins e objetos incorporados

Adobe AIR 1.0 e posterior

O AIR suporta o plug-in do Adobe® Acrobat®. Os usuários devem ter o Acrobat ou Adobe® Reader® 8.1 (ou superior) para exibir conteúdo PDF. O objeto HTMLLoader fornece uma propriedade para verificar se o sistema de um usuário pode exibir PDF. O conteúdo de arquivo SWF também pode ser exibido no ambiente HTML, mas esse recurso é incorporado ao AIR e não usa um plug-in externo.

Nenhum outro plug-in de Webkit é suportado no AIR.

Mais tópicos da Ajuda

"Segurança HTML no Adobe AIR" na página 1121

"Caixas de proteção HTML" na página 998

"Elementos HTML frame e iframe" na página 1005

"Objeto Window do JavaScript" na página 1003

"O objeto XMLHttpRequest" na página 999

"Inclusão de conteúdo em PDF no AIR" na página 568

AIR e Webkit

Adobe AIR 1.0 e posterior

O Adobe AIR usa o mecanismo do Webkit de código aberto, também usado no navegador da Web Safari. O AIR adiciona várias extensões para permitir acesso às classes e objetos de tempo de execução, bem como à segurança. Além disso, o Webkit por si só adiciona recursos não incluídos nos padrões da W3C para HTML, CSS e JavaScript.

Apenas as adições do AIR e as extensões do Webkit mais dignas de nota são tratadas aqui. Para obter uma documentação adicional sobre HTML não padrão, CSS e JavaScript, consulte www.webkit.org e developer.apple.com. Para obter informações sobre padrões, consulte o site da W3C. O Mozilla também fornece uma referência geral valiosa sobre tópicos HTML, CSS e DOM (é claro que os mecanismos do Webkit e da Mozilla não são idênticos).

JavaScript no AIR

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O AIR faz várias alterações no comportamento típico de objetos JavaScript comuns. Várias dessas alterações são feitas para tornar fácil escrever aplicativos seguros no AIR. Ao mesmo tempo, essas diferenças em comportamento significam que alguns padrões de codificação JavaScript comuns, e aplicativos Web existentes usando esses padrões, nem sempre podem executar conforme esperado no AIR. Para obter informações sobre como corrigir esses tipos de problemas, consulte "Como evitar erros JavaScript relacionados à segurança" na página 1016.

Caixas de proteção HTML

Adobe AIR 1.0 e posterior

O AIR coloca conteúdo em caixas de proteção isoladas de acordo com a origem do conteúdo. As regras da caixa de proteção são consistentes com a política de mesma origem implementada pela maioria dos navegadores da Web, bem como as regras para caixas de proteção implementadas pelo Adobe Flash Player. Além disso, o AIR fornece um novo tipo de caixa de proteção do *aplicativo* para conter e proteger conteúdo do aplicativo. Consulte "Caixas de proteção de segurança" na página 1081 para obter mais informações sobre os tipos de caixas de proteção que você pode encontrar ao desenvolver aplicativos AIR.

O acesso ao ambiente de tempo de execução e às APIs do AIR está disponível apenas para HTML e JavaScript em execução na caixa de proteção do aplicativo. Ao mesmo tempo, no entanto, a execução e avaliação dinâmica de JavaScript, em suas várias formas, são amplamente restritas na caixa de proteção do aplicativo por razões de segurança. Essas restrições são adequadas quer seu aplicativo realmente carregue ou não informações diretamente de um servidor. (Até mesmo conteúdo de arquivo, sequências de caracteres coladas e entrada do usuário direta podem ser incertos.)

A origem do conteúdo em uma página determina a caixa de proteção à qual ele é consignado. Apenas o conteúdo carregado do diretório do aplicativo (o diretório de instalação referenciado pelo `app:` esquema de URL) é colocado na caixa de proteção do aplicativo. O conteúdo carregado do sistema de arquivos é colocado na caixa de proteção *local com sistema de arquivos* ou *confiável local*, que permite acesso e interação com conteúdo no sistema de arquivos local, mas não conteúdo remoto. O conteúdo carregado da rede é colocado em uma caixa de proteção remota correspondente ao seu domínio de origem.

Para permitir que uma página de aplicativo interaja livremente com o conteúdo em uma caixa de proteção remota, a página pode ser mapeada para o mesmo domínio do conteúdo remoto. Por exemplo, se você escreve um aplicativo que exibe dados de mapas de um serviço de Internet, a página do seu aplicativo que carrega e exibe conteúdo do serviço poderia ser mapeada para o domínio de serviço. Os atributos para mapear páginas em um domínio e uma caixa de proteção remota são novos atributos adicionados aos elementos HTML frame e iframe.

Para permitir conteúdo em uma caixa de proteção que não seja de aplicativo para usar com segurança recursos do AIR, você pode configurar uma ponte de caixa de proteção pai. Para permitir que o conteúdo de aplicativo chame com segurança métodos e acesse propriedades de conteúdo em outras caixas de proteção, você pode configurar uma ponte de caixa de proteção filha. Segurança aqui significa que o conteúdo remoto não pode obter acidentalmente referências a objetos, propriedades ou métodos que não são expostos explicitamente. Apenas tipos de dados simples, funções e objetos anônimos podem ser transmitidos pela ponte. No entanto, você ainda deve evitar expor explicitamente funções potencialmente perigosas. Se, por exemplo, você expôs uma interface que permitiu ao conteúdo remoto ler e escrever arquivos em qualquer lugar no sistema de um usuário, você pode estar fornecendo ao conteúdo remoto o meio para fazer um dano considerável aos seus usuários.

Função eval() do JavaScript

Adobe AIR 1.0 e posterior

O uso da função `eval()` é restrito à caixa de proteção do aplicativo depois que o carregamento de uma página tiver sido concluído. Alguns usos são permitidos para que dados formatados por JSON possam ser analisados com segurança, mas qualquer avaliação que resulte em instruções executáveis terá como consequência um erro. O capítulo "Restrições de código de conteúdo em caixas de proteção distintas" na página 1123 descreve os usos permitidos da função `eval()`.

Construtores de funções

Adobe AIR 1.0 e posterior

Na caixa de proteção do aplicativo, os construtores de funções podem ser usados antes que o carregamento de uma página tenha sido concluído. Após todos os manipuladores de eventos `load` da página terem sido concluídos, novas funções não podem ser criadas.

Carregamento de scripts externos

Adobe AIR 1.0 e posterior

As páginas HTML na caixa de proteção do aplicativo não podem usar a tag de `script` para carregar arquivos de JavaScript de fora do diretório do aplicativo. Para uma página no seu aplicativo carregar um script de fora do diretório do aplicativo, a página deve ser mapeada para uma caixa de proteção que não seja de aplicativo.

O objeto XMLHttpRequest

Adobe AIR 1.0 e posterior

O AIR fornece um objeto XMLHttpRequest (XHR) que os aplicativos podem usar para fazer solicitações de dados. O exemplo a seguir ilustra uma solicitação de dados simples:

```
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);
```

Em contraste com um navegador, o AIR permite que o conteúdo em execução na caixa de proteção do aplicativo solicite dados de qualquer domínio. O resultado de um XHR que contém uma sequência de caracteres JSON pode ser avaliado em objetos de dados, a menos que o resultado também contenha código executável. Se instruções executáveis estão presentes no resultado de XHR, um erro é lançado e a tentativa de avaliação falha.

Para impedir a injeção acidental de código de fontes remotas, os XHRs síncronos retornam um resultado vazio se feito antes que o carregamento de uma página seja concluído. Os XHRs assíncronos sempre são retornados após o carregamento de uma página.

Por padrão, o AIR bloqueia XMLHttpRequests entre vários domínios em caixas de proteção de não aplicativos. Uma janela pai na caixa de proteção do aplicativo pode optar por permitir solicitações entre domínios em um frame filho com conteúdo em uma caixa de proteção que não seja de aplicativo definindo `allowCrossDomainXHR`, um atributo adicionado pelo AIR, como `true` no elemento frame ou iframe contido:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    allowCrossDomainXHR="true"
</iframe>
```

Nota: *Quando conveniente, a classe URLStream do AIR também pode ser usada para fazer o download de dados.*

Se você despachar um XMLHttpRequest para um servidor remoto de um frame ou iframe contendo conteúdo de aplicativo que tenha sido mapeado para uma caixa de proteção remota, verifique se a URL de mapeamento não mascara o endereço do servidor usado no XHR. Por exemplo, considere a seguinte definição de iframe, que mapeia conteúdo de aplicativo em uma caixa de proteção remota para o domínio example.com:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/"
    allowCrossDomainXHR="true"
</iframe>
```

Como o atributo `sandboxRoot` remapeia a URL raiz do endereço www.example.com, todas as solicitações são carregadas do diretório do aplicativo, e não do servidor remoto. As solicitações são remapeadas derivem elas de navegação de página ou de um XMLHttpRequest.

Para evitar bloquear solicitações de dados acidentalmente para o seu servidor remoto, mapeie `sandboxRoot` para um subdiretório da URL remota, e não da raiz. O diretório não precisa existir. Por exemplo, para permitir solicitações ao www.example.com para ser carregado do servidor remoto, e não do diretório do aplicativo, altere o iframe anterior para o seguinte:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/air/"
    allowCrossDomainXHR="true"
</iframe>
```

Nesse caso, apenas conteúdo no subdiretório `air` é carregado localmente.

Para obter mais informações sobre o mapeamento de caixa de proteção, consulte "Elementos HTML frame e iframe" na página 1005 e "Segurança HTML no Adobe AIR" na página 1121.

Cookies

Adobe AIR 1.0 e posterior

Em aplicativos AIR, apenas o conteúdo em caixas de proteção remotas (conteúdo carregado de fontes http: e https:) pode usar cookies (a propriedade `document.cookie`). Na caixa de proteção do aplicativo, há outras formas para armazenar os dados persistentes, incluindo as classes EncryptedLocalStore, SharedObject e FileStream.

O objeto Clipboard

Adobe AIR 1.0 e posterior

A API Clipboard do WebKit é conduzida com os seguintes eventos: `copy`, `cut` e `paste`. O objeto event transmitido nesses eventos fornece acesso à área de transferência pela propriedade `clipboardData`. Use os seguintes métodos do objeto `clipboardData` para ler ou escrever dados da área de transferência:

Método	Descrição
clearData(mimeType)	Limpa os dados da área de transferência. Defina o parâmetro `mimeType` para o tipo MIME dos dados a apagar.
getData(mimeType)	Obtém os dados da área de transferência. Esse método pode ser chamado apenas em um manipulador para o evento `paste`. Defina o parâmetro `mimeType` para o tipo MIME dos dados a retornar.
setData(mimeType, data)	Copia dados para a área de transferência. Defina o parâmetro `mimeType` para o tipo MIME dos dados.

O código JavaScript fora da caixa de proteção do aplicativo pode acessar apenas a área de transferência por esses eventos. No entanto, o conteúdo na caixa de proteção do aplicativo pode acessar a área de transferência do sistema diretamente usando a classe Clipboard do AIR. Por exemplo, você poderia usar a seguinte instrução para obter dados do formato do texto na área de transferência:

```
var clipping = air.Clipboard.generalClipboard.getData("text/plain",
                                air.ClipboardTransferMode.ORIGINAL_ONLY);
```

Os tipos MIME de dados válidos são:

Tipo MIME	Valor
Texto	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
Lista de arquivos	"application/x-vnd.adobe.air.file-list"

Importante: *Apenas conteúdo na caixa de proteção do aplicativo pode acessar dados de arquivos presentes na área de transferência. Se um conteúdo que não seja de aplicativo tentar acessar um objeto de arquivo da área de transferência, será lançado um erro de segurança.*

Para obter mais informações sobre o uso da área de transferência, consulte "Copiar e colar" na página 614 e Utilização da área de trabalho de JavaScript (Centro de desenvolvedores da Apple).

Arrastar e soltar

Adobe AIR 1.0 e posterior

Gestos de arrastar e soltar para dentro e para fora do HTML produzem os seguintes eventos DOM: `dragstart`, `drag`, `dragend`, `dragenter`, `dragover`, `dragleave` e `drop`. O objeto event transmitido nesses eventos fornece acesso aos dados arrastados pela propriedade `dataTransfer`. A propriedade `dataTransfer` faz referência a um objeto que fornece os mesmos métodos do objeto `clipboardData` associado a um evento clipboard. Por exemplo, você poderia usar a seguinte função para obter dados do formato de um evento `drop`:

```
function onDrop(dragEvent){
    return dragEvent.dataTransfer.getData("text/plain",
            air.ClipboardTransferMode.ORIGINAL_ONLY);
}
```

O objeto `dataTransfer` possui os seguintes membros importantes:

Membro	Descrição
clearData(mimeType)	Limpa os dados. Defina o parâmetro `mimeType` para o tipo MIME da representação de dados para apagar.
getData(mimeType)	Obtém os dados arrastados. Esse método pode ser chamado apenas em um manipulador para o evento `drop`. Defina o parâmetro `mimeType` para o tipo MIME dos dados a obter.
setData(mimeType, data)	Defina os dados a serem arrastados. Defina o parâmetro `mimeType` para o tipo MIME dos dados.
tipos	Uma matriz de sequências de caracteres contendo os tipos MIME de todas as representações de dados disponíveis no momento no objeto `dataTransfer`.
effectsAllowed	Especifica se os dados que estão sendo arrastados podem ser copiados, movidos, vinculados ou alguma combinação disso. Defina a propriedade `effectsAllowed` no manipulador para o evento `dragstart`.
dropEffect	Especifica quais dos efeitos drop permitidos são suportados por um drag target. Defina a propriedade `dropEffect` no manipulador para o evento `dragEnter`. Durante a ação de arrastar, o cursor muda para indicar que efeito ocorreria se o usuário soltasse o mouse. Se nenhum `dropEffect` for especificado, um efeito da propriedade `effectsAllowed` será escolhido. O efeito de copiar tem prioridade sobre o efeito de mover, que por si só tem prioridade sobre o efeito de vincular. O usuário pode modificar a prioridade padrão usando o teclado.

Para obter mais informações sobre como adicionar suporte para arrastar e soltar a um aplicativo do AIR, consulte "Arrastar e soltar no AIR" na página 627 e Utilização de arrastar e soltar de JavaScript (Centro de desenvolvedores da Apple).

Propriedades innerHTML e outerHTML

Adobe AIR 1.0 e posterior

O AIR coloca restrições de segurança sobre o uso das propriedades `innerHTML` e `outerHTML` para conteúdo em execução na caixa de proteção do aplicativo. Antes do evento de carregamento da página, bem como durante a execução de qualquer manipulador de evento de carregamento, o uso das propriedades `innerHTML` e `outerHTML` é irrestrito. No entanto, depois que a página é carregada, você pode apenas usar as propriedades `innerHTML` ou `outerHTML` para adicionar conteúdo estático ao documento. Qualquer instrução na sequência de caracteres atribuída a `innerHTML` ou `outerHTML` avaliada como código executável é ignorada. Por exemplo, se você incluir um atributo de retorno de evento em uma definição de elemento, o ouvinte de eventos não será adicionado. Da mesma forma, tags `<script>` incorporadas não são avaliadas. Para obter mais informações, consulte "Segurança HTML no Adobe AIR" na página 1121.

Métodos Document.write() e Document.writeln()

Adobe AIR 1.0 e posterior

O uso dos métodos `write()` e `writeln()` não é restrito à caixa de proteção do aplicativo antes do evento `load` da página. No entanto, depois que a página é carregada, chamar um desses métodos não limpa a página ou cria uma nova. Em uma caixa de proteção que não seja de aplicativo, como na maioria dos navegadores da Web, chamar `document.write()` ou `writeln()`, após a conclusão do carregamento de uma página, limpa a página atual e abre uma nova em branco.

Propriedade Document.designMode

Adobe AIR 1.0 e posterior

Defina a propriedade `document.designMode` para um valor de `on` para tornar todos os elementos do documento editáveis. O suporte a editor incorporado inclui editar texto, copiar, colar e arrastar e soltar. Definir `designMode` como `on` é equivalente a definir a propriedade `contentEditable` do elemento `body` como `true`. Você pode usar a propriedade `contentEditable` na maioria dos elementos HTML para definir que seções de um documento são editáveis. Consulte "Atributo contentEditable HTML" na página 1008 para obter informações adicionais.

Eventos unload (para objetos body e frameset)

Adobe AIR 1.0 e posterior

Na tag de nível superior `frameset` ou `body` de uma janela (incluindo a janela principal do aplicativo), não use o evento `unload` para responder à janela (ou aplicativo) que está sendo fechada. Em vez disso, use o evento `exiting` do objeto NativeApplication (para detectar quando um aplicativo está sendo fechando). Em vez disso, use o evento `exiting` do objeto NativeWindow (para detectar quando um aplicativo está sendo fechado). Por exemplo, o seguinte código JavaScript exibe uma mensagem (`"Goodbye."`) quando o usuário fecha o aplicativo:

```
var app = air.NativeApplication.nativeApplication;
app.addEventListener(air.Event.EXITING, closeHandler);
function closeHandler(event)
{
    alert("Goodbye.");
}
```

No entanto, scripts *podem* responder com êxito ao evento `unload` causado pela navegação de um conteúdo frame, iframe ou window de nível superior.

Nota: *Essas limitações podem ser removidas em uma versão futura do Adobe AIR.*

Objeto Window do JavaScript

Adobe AIR 1.0 e posterior

O objeto Window permanece o objeto global no contexto de execução do JavaScript. Na caixa de proteção do aplicativo, o AIR adiciona novas propriedades ao objeto Window do JavaScript para fornecer acesso às classes incorporadas do AIR, bem como importantes objetos de host. Além disso, alguns métodos e propriedades se comportam de modo diferente dependendo se estão ou não na caixa de proteção do aplicativo.

Propriedade Window.runtime A propriedade `runtime` permite que você instancie e use classes runtime incorporadas da caixa de proteção do aplicativo. Essas classes incluem APIs do AIR e do Flash Player (mas não, por exemplo, a estrutura do Flex). Por exemplo, a seguinte instrução cria um objeto de arquivo do AIR:

```
var preferencesFile = new window.runtime.flash.filesystem.File();
```

O arquivo `AIRAliases.js`, fornecido no SDK do AIR, contém definições de aliases que permitem a você encurtar tais referências. Por exemplo, quando `AIRAliases.js` é importado em uma página, um objeto File pode ser criado com a seguinte instrução:

```
var preferencesFile = new air.File();
```

A propriedade `window.runtime` somente é definida para conteúdo dentro da caixa de proteção do aplicativo e apenas para o documento pai de uma página com frames ou iframes.

Consulte "Uso do arquivo AIRAliases.js" na página 1022.

Propriedade Window.nativeWindow A propriedade `nativeWindow` fornece uma referência ao objeto de janela nativa subjacente. Com essa propriedade, você pode fazer o script de funções e propriedades de janela, como posição da tela, tamanho e visibilidade e manipular eventos de janela, como fechar, redimensionar e mover. Por exemplo, a seguinte instrução fecha a janela:

```
window.nativeWindow.close();
```

Nota: *Os recursos de controle da janela fornecidos pelo objeto NativeWindow sobrepõem os recursos fornecidos pelo objeto Window do JavaScript. Em tais casos, você pode usar qualquer método que considerar mais conveniente.*

A propriedade `window.nativeWindow` somente é definida para conteúdo dentro da caixa de proteção do aplicativo e apenas para o documento pai de uma página com frames ou iframes.

Propriedade Window.htmlLoader A propriedade `htmlLoader` fornece uma referência ao objeto AIR HTMLLoader que contém o conteúdo HTML. Com essa propriedade, você pode fazer o script da aparência e do comportamento do ambiente HTML. Por exemplo, você pode usar a propriedade `htmlLoader.paintsDefaultBackground` para determinar se o controle pinta um plano de fundo branco padrão:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Nota: *O objeto HTMLLoader por si só possui uma propriedade `window`, que faz referência ao objeto Window do JavaScript do conteúdo HTML contido nele. Você pode usar essa propriedade para acessar o ambiente JavaScript por uma referência ao HTMLLoader contido.*

A propriedade `window.htmlLoader` somente é definida para conteúdo dentro da caixa de proteção do aplicativo e apenas para o documento pai de uma página com frames ou iframes.

Propriedades Window.parentSandboxBridge e Window.childSandboxBridge As propriedades `parentSandboxBridge` e `childSandboxBridge` permitem que você defina uma interface entre um frame filho e um pai. Para obter mais informações, consulte "Conteúdo cross-scripting em caixas de proteção de segurança distintas" na página 1033.

Funções Window.setTimeout() e Window.setInterval() O AIR coloca restrições de segurança sobre o uso das funções `setTimeout()` e `setInterval()` na caixa de proteção do aplicativo. Não é possível definir o código a ser executado como uma sequência de caracteres ao chamar `setTimeout()` ou `setInterval()`. É necessário usar uma referência de função. Para obter mais informações, consulte "setTimeout() e setInterval()" na página 1019.

Função Window.open() Quando chamado por código em execução em uma caixa de proteção que não seja de aplicativo, o método `open()` abrirá somente uma janela quando chamado como resultado da interação do usuário (como um clique do mouse ou pressionamento de tecla). Além disso, o título da janela é prefixado com o título do aplicativo (para impedir que janelas sejam abertas por conteúdo remoto personifiquem janelas abertas pelo aplicativo). Para obter mais informações, consulte o capítulo "Restrições na chamada do método window.open() de JavaScript" na página 1127.

Objeto air.NativeApplication

Adobe AIR 1.0 e posterior

O objeto NativeApplication fornece informações sobre o estado do aplicativo, despacha vários eventos de nível de aplicativo importantes e fornece funções úteis para controlar o comportamento do aplicativo. Uma única instância do objeto NativeApplication é criada automaticamente e pode ser acessada pela propriedade `NativeApplication.nativeApplication` definida pela classe.

Para acessar o objeto do código JavaScript, você poderia usar:

```
var app = window.runtime.flash.desktop.NativeApplication.nativeApplication;
```

Ou, se o script `AIRAliases.js` tiver sido importado, você poderia usar a forma mais curta:

```
var app = air.NativeApplication.nativeApplication;
```

O objeto NativeApplication pode apenas ser acessado de dentro da caixa de proteção do aplicativo. Para obter mais informações sobre o objeto NativeApplication, consulte "Trabalho com informações de tempo de execução do AIR e de sistema operacional" na página 916.

O esquema de URL JavaScript

Adobe AIR 1.0 e posterior

A execução do código definida em um esquema de URL de JavaScript (como em `href="javascript:alert('Test')"`) é bloqueada na caixa de proteção do aplicativo. Nenhum erro é lançado.

HTML no AIR

Adobe AIR 1.0 e posterior

O AIR e o WebKit definem alguns atributos e elementos HTML não padrão, incluindo:

"Elementos HTML frame e iframe" na página 1005

"Manipuladores de eventos de elementos HTML" na página 1007

Elementos HTML frame e iframe

Adobe AIR 1.0 e posterior

O AIR adiciona novos atributos aos elementos frame e iframe de conteúdo na caixa de proteção do aplicativo:

Atributo sandboxRoot O atributo `sandboxRoot` especifica um domínio de origem alternativo que não seja de aplicativo para o arquivo especificado pelo atributo `src` do frame. O arquivo é carregado na caixa de proteção que não seja de aplicativo correspondente ao domínio especificado. O conteúdo no arquivo e o conteúdo carregado do domínio especificado podem cruzar scripts entre si.

Importante: *Se você definir o valor de `sandboxRoot` para a URL base do domínio, todas as solicitações para conteúdo desse domínio serão carregadas do diretório do aplicativo, e não do servidor remoto (resulte a solicitação de navegação de página, de um XMLHttpRequest ou de qualquer outro meio de carregar conteúdo).*

Atributo documentRoot O atributo `documentRoot` especifica o diretório local a partir do qual carregar URLs que procuram endereços para arquivos no local especificado por `sandboxRoot`.

Ao consultar endereços de URLs, no atributo `src` do frame ou no conteúdo carregado no frame, a parte da URL que corresponde ao valor especificado em `sandboxRoot` é substituída pelo valor especificado em `documentRoot`. Portanto, na seguinte tag de frame:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

`child.html` é carregado do subdiretório `sandbox` da pasta de instalação do aplicativo. Os endereços das URLs relativas em `child.html` são procurados com base no diretório `sandbox`. Observe que qualquer arquivo no servidor remoto em `www.example.com/air` não pode ser acessado no frame, uma vez que o AIR tentaria carregá-lo do diretório app:/sandbox/.

Atributo allowCrossDomainXHR Inclua `allowCrossDomainXHR="allowCrossDomainXHR"` na tag de frame de abertura para permitir que o conteúdo no frame faça XMLHttpRequests para qualquer domínio remoto. Por padrão, o conteúdo que não é de aplicativo apenas pode fazer tais solicitações em seu próprio domínio de origem. Existem implicações de segurança sérias envolvidas na permissão de XHRs entre domínios. O código da página pode trocar dados com qualquer domínio. Se o conteúdo mal-intencionado é de certa forma injetado na página, qualquer dado acessível para código na caixa de proteção atual pode ficar comprometido. Apenas ative XHRs entre domínios para páginas que você cria e controla e somente quando o carregamento de dados entre domínios for realmente necessário. Além disso, valide cuidadosamente todos os dados externos carregados pela página para impedir a injeção de dados ou outras formas de ataque.

Importante: *Se o atributo `allowCrossDomainXHR` for incluído em um elemento frame ou iframe, XHRs entre domínios serão ativados (a menos que o valor atribuído seja "0" ou comece com as letras "f" ou "n"). Por exemplo, definir `allowCrossDomainXHR` como `"deny"` ainda ativaria XHRs entre domínios. Deixe o atributo totalmente fora da declaração do elemento se não desejar ativar solicitações entre domínios.*

Atributo ondominitialize Especifica um manipulador de eventos para o evento `dominitialize` de um frame. Esse é um evento específico do AIR disparado quando os objetos window e document do frame tiverem sido criados, mas antes que qualquer script tenha sido analisado ou elementos document tenham sido criados.

O frame despacha o evento `dominitialize` cedo o suficiente na sequência de carregamento de forma que qualquer script na página filha possa se referir a objetos, variáveis e funções adicionadas ao documento filho pelo manipulador `dominitialize`. A página pai deve estar na mesma caixa de proteção da filha para adicionar ou acessar diretamente qualquer objeto em um documento filho. No entanto, um pai na caixa de proteção do aplicativo pode estabelecer uma ponte de caixa de proteção para se comunicar com conteúdo em uma caixa de proteção que não seja de aplicativo.

Os exemplos a seguir ilustram o uso da tag iframe no AIR:

Coloque `child.html` em uma caixa de proteção remota, sem mapear para um domínio real em um servidor remoto:

```
<iframe src="http://localhost/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://localhost/air/"/>
```

Coloque `child.html` em uma caixa de proteção remota, permitindo XMLHttpRequests apenas para `www.example.com`:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

Coloque `child.html` em uma caixa de proteção remota, permitindo XMLHttpRequests para qualquer domínio remoto:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"
            allowCrossDomainXHR="allowCrossDomainXHR"/>
```

Coloque `child.html` em uma caixa de proteção local com sistema de arquivos:

```
<iframe     src="file:///templates/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="app-storage:/templates/"/>
```

Coloque `child.html` em uma caixa de proteção remota, usando o evento `dominitialize` para estabelecer uma ponte de caixa de proteção:

```
<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>
```

O seguinte documento `child.html` ilustra como o conteúdo filho pode acessar a ponte de caixa de proteção pai:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Para obter mais informações, consulte "Conteúdo cross-scripting em caixas de proteção de segurança distintas" na página 1033 e "Segurança HTML no Adobe AIR" na página 1121.

Manipuladores de eventos de elementos HTML

Adobe AIR 1.0 e posterior

Os objetos DOM no AIR e no Webkit despacham alguns eventos não encontrados no modelo de eventos DOM padrão. A tabela a seguir lista os atributos de eventos relacionados que podem ser usados para especificar manipuladores para estes eventos:

Nome do atributo de retorno de chamada	Descrição
oncontextmenu	Chamado quando um menu de contexto é invocado, como pelo clique do botão direito do mouse ou clique da tecla Command no texto selecionado.
oncopy	Chamado quando uma seleção em um elemento é copiada.
oncut	Chamado quando uma seleção em um elemento é cortada.
ondominitialize	Chamado quando o DOM de um documento carregado em um frame ou iframe é criado, mas antes que qualquer elemento de DOM seja criado ou script analisado.
ondrag	Chamado quando um elemento é arrastado.
ondragend	Chamado quando uma ação de arrastar é liberada.
ondragenter	Chamado quando um gesto de ação de arrastar entra nos limites de um elemento.
ondragleave	Chamado quando um gesto de ação de arrastar deixa os limites de um elemento.
ondragover	Chamado continuamente enquanto um gesto de ação de arrastar está dentro dos limites de um elemento.
ondragstart	Chamado quando um gesto de ação de arrastar é iniciado.
ondrop	Chamado quando um gesto de ação de arrastar é liberado sobre um elemento.
onerror	Chamado quando um erro ocorre ao carregar um elemento.
oninput	Chamado quanto o texto é inserido em um elemento de formulário.
onpaste	Chamado quando um item é colado em um elemento.
onscroll	Chamado quando o conteúdo de um elemento rolável é rolado.
onselectstart	Chamado quando uma seleção é iniciada.

Atributo contentEditable HTML

Adobe AIR 1.0 e posterior

Você pode adicionar o atributo `contentEditable` a qualquer elemento HTML para permitir aos usuários editar o conteúdo do elemento. Por exemplo, o seguinte código de exemplo HTML define todo o documento como editável, exceto para o primeiro elemento `p`:

```
<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>
```

Nota: *Se você define a propriedade* `document.designMode` *como* `on`*, todos os elementos do documento são editáveis, independentemente da configuração de* `contentEditable` *para um elemento individual. No entanto, definir* `designMode` *como* `off` *não desativa a edição de elementos para os quais* `contentEditable` *é* `true`*. Consulte* *"Propriedade Document.designMode" na página 1003* *para obter informações adicionais.*

Dados: URLs

Adobe AIR 2 e posterior

O AIR oferece suporte a URLs `data:` para os seguintes elementos:

- img
- tipo de entrada ="imagem"
- Regras CSS que permitem imagens (por exemplo: imagens de fundo)

URLs de dados permitem que você insira dados de imagens binárias diretamente em um documento CSS ou HTML como em uma string codificada em base 64. O exemplo a seguir utiliza uma URL:dados como um segundo plano repetitivo.

```
<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>
```

Ao utilizar URLS: dados, saiba que espaços em branco extra são insignificantes. Por exemplo, a string de dados precisa ser inserida como uma linha única e sem quebra. Caso contrário, as quebras de linha serão tratadas como parte dos dados e a imagem não poderá ser decodificada.

CSS no AIR

Adobe AIR 1.0 e posterior

O WebKit suporta várias propriedades CSS estendidas. Muitas destas extensões utilizam o prefixo `-webkit`. Observe que algumas destas extensões são de natureza experimental e podem ser removidas de uma versão futura do WebKit. Para obter mais informações sobre o suporte do WebKit para CSS e de suas extensões para p CSS, consulte Referência do Safari CSS.

Recursos do WebKit não suportados no AIR

Adobe AIR 1.0 e posterior

O AIR não oferece suporte aos seguintes recursos disponíveis no WebKit ou no Safari 4:

- Mensagens entre domínios por meio de window.postMessage (o AIR fornece suas próprias APIs de comunicação entre domínios)
- Variáveis CSS
- Fontes WOFF (Web Open Font Format) e SVG.
- Tags HTML de vídeo e áudio
- Consultas de dispositivos de mídia
- Cache de aplicativos offline
- Imprimindo (O AIR fornece sua própria API PrintJob)
- Verificadores de ortografia e gramática
- SVG
- WAI-ARIA
- WebSockets (O AIR fornece suas próprias APIs de soquete)
- Web workers
- API SQL do WebKit (o AIR fornece sua própria API)
- API de geolocalização do WebKit (o AIR fornece sua própria API de geolocalização em dispositivos suportados)
- API para upload de vários arquivos
- Eventos de toque do WebKit (O AIR fornece seus próprios eventos de toque)
- Wireless Markup Language (WML)

As listas a seguir contém APIs JavaScript específicas, elementos HTML e propriedades CSS, e valores que o AIR não suporta:

Membros do objeto JavaScript Window não suportados:

- applicationCache()
- console
- openDatabase()
- postMessage()
- document.print()

Tags HTML não suportadas:

- áudio
- video

Atributos HTML não suportados:

- aria-*
- draggable

- formnovalidate
- list
- novalidate
- onbeforeload
- onhashchange
- onorientationchange
- onpagehide
- onpageshow
- onpopstate
- ontouchstart
- ontouchmove
- ontouchend
- ontouchcancel
- onwebkitbeginfullscreen
- onwebkitendfullscreen
- pattern
- required
- sandbox

Eventos JavaScript não suportados:

- beforeload
- hashchange
- orientationchange
- pagehide
- pageshow
- popstate
- touchstart
- touchmove
- touchend
- touchcancel
- webkitbeginfullscreen
- webkitendfullscreen

Propriedades CSS não suportadas:

- background-clip
- background-origin (uso -webkit-background-origin)
- background-repeat-x
- background-repeat-y

- background-size (uso -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

Valores CSS não suportados:

- valores de propriedade de aparência:
  - media-volume-slider-container
  - media-volume-slider
  - media-volume-sliderthumb
  - outer-spin-button
- border-box (background-clip e background-origin)
- contain (background-size)
- content-box (background-clip e background-origin)
- cover (background-size)
- valores de lista de propriedade:
  - 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 (propriedade display)

Capítulo 59: Programação de HTML e JavaScript no AIR

Adobe AIR 1.0 e posterior

Vários tópicos de programação são exclusivos para o desenvolvimento de aplicativos Adobe® AIR® com HTML e JavaScript. As seguintes informações são importantes caso você esteja programando um aplicativo baseado em HTML ou um aplicativo do AIR baseado em SWF que execute HTML e JavaScript usando a classe HTMLLoader (ou o componente mx:HTML Flex™).

Sobre a classe HTMLLoader

Adobe AIR 1.0 e posterior

A classe HTMLLoader do Adobe AIR define o objeto de exibição que pode mostrar conteúdo HTML em um aplicativo do AIR. Os aplicativos baseados em SWF podem adicionar um controle HTMLLoader a uma janela existente ou criar uma janela HTML que contenha automaticamente um objeto HTMLLoader com `HTMLLoader.createRootWindow()`. O objeto HTMLLoader pode ser acessado por meio da propriedade `window.htmlLoader` de JavaScript de dentro da página HTML carregada.

Carregamento de conteúdo HTML de uma URL

Adobe AIR 1.0 e posterior

O código a seguir carrega uma URL no objeto HTMLLoader (adicione o HTMLLoader como filho do estágio ou outro contêiner do objeto de exibição para exibir o conteúdo HTML no seu aplicativo):

```
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);
```

Por padrão, as propriedades `width` e `height` de um objeto HTMLLoader são definidas como 0. Você desejará definir essas dimensões quando adicionar um objeto HTMLLoader ao palco. O HTMLLoader despacha diversos eventos à medida que a página carrega. Você pode usar esses eventos para determinar quando é seguro interagir com a página carregada. Esses eventos são descritos em "Tratamento de eventos relacionados a HTML no AIR" na página 1057.

Nota: *Na estrutura do Flex, somente as classes que estendem a classe UIComponent podem ser adicionadas como filhos de componentes do recipiente Flex. Por esse motivo, você não pode adicionar diretamente o HTMLLoader como filho de um componente do recipiente do Flex; no entanto, é possível usar o controle Flex mx:HTML; você pode criar uma classe personalizada que estende UIComponent e contém um HTMLLoader como filho do UIComponent ou pode adicionar o HTMLLoader como filho de um UIComponent e adicionar o UIComponent ao recipiente do Flex.*

Você também pode processar texto HTML usando a classe TextField, mas os respectivos recursos são limitados. A classe Textfield do Adobe® Flash® Player oferece suporte a um subconjunto de markup de HTML, mas, devido a limitações de tamanho, os recursos respectivos são limitados. (A classe HTMLLoader incluída no Adobe AIR não está disponível no Flash Player.)

Carregamento de conteúdo HTML de uma sequência de caracteres

Adobe AIR 1.0 e posterior

O método `loadString()` do objeto HTMLLoader carrega uma string de conteúdo HTML no objeto HTMLLoader:

```
var html:HTMLLoader = new HTMLLoader();
var htmlStr:String = "<html><body>Hello <b>world</b>.</body></html>";
html.loadString(htmlStr);
```

Por padrão, o conteúdo carregado por meio do método `loadString()` é colocado em uma caixa de proteção que não é aplicativo com as seguintes características:

- Ela tem acesso ao conteúdo carregado da rede (mas não do sistema de arquivos).
- Ela não pode carregar dados usando XMLHttpRequest.
- A propriedade `window.location` é definida como `"about:blank"`.
- O conteúdo não pode acessar a propriedade `window.runtime` (como pode em qualquer caixa de proteção que não é aplicativo).

No AIR 1.5, a classe HTMLLoader inclui uma propriedade `placeLoadStringContentInApplicationSandbox`. Quando essa propriedade é definida como `true` para um objeto HTMLLoader, o conteúdo carregado pelo método `loadString()` é colocado na caixa de proteção do aplicativo. (O valor padrão é `false`.) Isso fornece ao conteúdo carregado pelo método `loadString()` acesso à propriedade `window.runtime` e a todas as APIs do ARI. Se você definir essa propriedade como `true`, assegure que a fonte de dados para uma sequência de caracteres usada em uma chamada para o método `loadString()` seja confiável. As instruções de código na sequência de caracteres HTML são executadas com privilégios totais de aplicativo quando essa propriedade é definida como `true`. Só defina a propriedade como `true` quando estiver certo de que a sequência de caracteres não possa conter código nocivo.

Em aplicativos compilados com SDKs do AIR 1.0 ou 1.1, o conteúdo carregado pelo método `loadString()` é inserido na caixa de proteção do aplicativo.

Regras de segurança importantes no uso de HTML em aplicativos AIR

Adobe AIR 1.0 e posterior

Os arquivos instalados com o aplicativo do AIR têm acesso às respectivas APIs. Por motivos de segurança, o conteúdo de outras fontes não tem acesso. Por exemplo, essa restrição impede que conteúdo de um domínio remoto (como http://example.com) leia o conteúdo de diretório da área de trabalho do usuário (ou algo pior).

Como há buracos de segurança que podem ser explorados através da chamada da função `eval()` (e APIs relacionadas), o conteúdo instalado com o aplicativo, por padrão, não pode usar esses métodos. No entanto, algumas estruturas Ajax usam a chamada da função `eval()` e APIs relacionadas.

Para estruturar adequadamente o conteúdo para trabalhar em um aplicativo do AIR, você deve considerar as regras das restrições de segurança sobre conteúdo de fontes diversas. O conteúdo de fontes diversas é colocado em classificações de segurança distintas, chamadas de caixas de proteção (consulte"Caixas de proteção de segurança" na página 1081). Por padrão, o conteúdo instalado com o aplicativo está instalado em uma caixa de proteção conhecida como caixa de proteção de *aplicativo* e isso concede a ele acesso às APIs do AIR. Normalmente, a caixa de proteção do aplicativo é a mais segura, com restrições projetadas para impedir a execução de código não confiável.

O tempo de execução permite carregar o conteúdo instalado com o aplicativo em uma caixa de proteção diferente da caixa de proteção do aplicativo. O conteúdo em caixas de proteção que não são de aplicativo opera em um ambiente de segurança semelhante ao de um navegador da Web típico. Por exemplo, o código das caixas de proteção que não são de aplicativo pode usar `eval()` e métodos relacionados (mas, ao mesmo tempo, não é permitido acesso às APIs do AIR). O tempo de execução inclui maneiras de fazer com que o conteúdo em caixas de proteção distintas se comuniquem com segurança (sem expor, por exemplo, as APIs do AIR a conteúdo de não aplicativo). Para obter detalhes, consulte "Conteúdo cross-scripting em caixas de proteção de segurança distintas" na página 1033.

Se você chamar um código com uso restrito em uma caixa de proteção por motivos de segurança, o tempo de execução despachará um erro JavaScript: "Violação de segurança de tempo de execução do Adobe AIR para código JavaScript na caixa de proteção de segurança do aplicativo."

Para evitar esse erro, siga as práticas de codificação descritas na próxima seção, "Como evitar erros JavaScript relacionados à segurança" na página 1016.

Para obter mais informações, consulte "Segurança HTML no Adobe AIR" na página 1121.

Como evitar erros JavaScript relacionados à segurança

Adobe AIR 1.0 e posterior

Se você chamar um código com uso restrito em uma caixa de proteção, devido a essas restrições de segurança, o tempo de execução despachará um erro de JavaScript: "Violação de segurança de tempo de execução do Adobe AIR para código JavaScript na caixa de proteção de segurança do aplicativo." Para evitar esse erro, siga essas práticas de codificação.

Causas de erros JavaScript relacionados à segurança

Adobe AIR 1.0 e posterior

O código em execução na caixa de proteção do aplicativo está restrito à maioria das operações que envolvem avaliação e execução de sequências depois que o evento `load` tiver sido disparado e todos os manipuladores de evento `load` tiverem encerrado. A tentativa de usar os seguintes tipos de instruções JavaScript que avaliam e executam strings potencialmente inseguras gera erros JavaScript:

- função eval()
- setTimeout() e setInterval()
- Construtor de funções

  Além disso, os tipos de instruções JavaScript seguintes falham sem gerar um erro JavaScript inseguro:

- javascript: URLs
- Retornos de chamada de evento atribuídos por meio de atributos onevent em instruções innerHTML e outerHTML

- Carregamento de arquivos JavaScript externos ao diretório de instalação do aplicativo
- document.write() e document.writeln()
- XMLHttpRequests síncronas antes do evento load ou durante um manipulador de eventos load
- Elementos de script criados dinamicamente

  Nota: *Em alguns casos restritos, a avaliação de strings é permitida. Consulte "Restrições de código de conteúdo em caixas de proteção distintas" na página 1123 para obter mais informações.*

  A Adobe mantém uma lista das estruturas Ajax conhecidas para oferecer suporte à caixa de proteção de segurança do aplicativo em http://www.adobe.com/go/airappsandboxframeworks_br.

  As seções seguintes descrevem como regravar os scripts para evitar esses erros JavaScript inseguro e falhas silenciosas do código em execução na caixa de proteção do aplicativo.

Mapeamento de conteúdo do aplicativo para uma caixa de proteção distinta

Adobe AIR 1.0 e posterior

Na maioria dos casos, você pode regravar ou reestruturar o aplicativo para evitar erros JavaScript relacionados à segurança. No entanto, quando não for possível regra ou reestruturar, você poderá carregar o conteúdo do aplicativo em uma caixa de proteção distinta usando a técnica descrita em "Carregamento de conteúdo do aplicativo em uma caixa de proteção "não aplicativo"" na página 1034. Se esse conteúdo também deve acessar as APIs do AIR, você pode criar uma ponte de caixa de proteção, conforme descrito em "Configuração de interface de ponte de caixa de proteção" na página 1034.

função eval()

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Na caixa de proteção do aplicativo, a função `eval()` só pode ser usada antes do evento `load` da página ou durante o manipulador de eventos `load`. Após a página ter sido carregada, as chamadas para `eval()` não executarão código. No entanto, nos seguintes casos, você pode regravar o código para evitar o uso de `eval()`.

Atribuição de propriedades a um objeto

Adobe AIR 1.0 e posterior

Em vez de analisar uma string para criar o acessador de propriedades:

```
eval("obj." + propName + " = " + val);
```

acesse propriedades com notação entre colchetes:

```
obj[propName] = val;
```

Criação de função com variáveis disponíveis no contexto

Adobe AIR 1.0 e posterior

Substitua instruções como as seguintes:

```
function compile(var1, var2){
    eval("var fn = function(){ this."+var1+"(var2) }");
    return fn;
}
```

com:

```
function compile(var1, var2){
    var self = this;
    return function(){ self[var1](var2) };
}
```

Criação de objeto usando o nome da classe como parâmetro de sequência

Adobe AIR 1.0 e posterior

Considere uma classe JavaScript hipotética definida com o seguinte código:

```
var CustomClass =
    {
        Utils:
        {
            Parser: function(){ alert('constructor') }
        },
        Data:
        {

        }
    };
var constructorClassName = "CustomClass.Utils.Parser";
```

A maneira mais simples de criar uma ocorrência é usar `eval()`:

```
var myObj;
eval('myObj=new ' + constructorClassName +'()')
```

No entanto, você pode evitar a chamada para `eval()`, analisando cada componente de nome de classe e criando o novo objeto usando notação entre colchetes:

```
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;
}
```

Para criar a ocorrência, use:

```
try{
    var Parser = getter(constructorClassName);
    var a = new Parser();
    }catch(e){
        alert(e);
}
```

setTimeout() e setInterval()

Adobe AIR 1.0 e posterior

Substitua a string passada como a função do manipulador por uma referência de função ou objeto. Por exemplo, substitua uma instrução como a seguinte:

```
setTimeout("alert('Timeout')", 100);
```

com:

```
setTimeout(function(){alert('Timeout')}, 100);
```

Ou, quando a função precisar que o objeto `this` seja definido pelo chamador, substitua uma instrução como:

```
this.appTimer = setInterval("obj.customFunction();", 100);
```

pelo seguinte:

```
var _self = this;
this.appTimer = setInterval(function(){obj.customFunction.apply(_self);}, 100);
```

Construtor de funções

Adobe AIR 1.0 e posterior

As chamadas para `new Function(param, body)` podem ser substituídas por uma declaração de função inline ou usadas apenas antes que o evento `load` da página tenha sido tratado.

javascript: URLs

Adobe AIR 1.0 e posterior

O código definido em um link usando o javascript: O esquema de URL é ignorado na caixa de proteção do aplicativo. Não é gerado nenhum erro JavaScript inseguro. Você pode substituir links usando javascript: URLs, como:

```
<a href="javascript:code()">Click Me</a>
```

com:

```
<a href="#" onclick="code()">Click Me</a>
```

Retornos de chamada de evento atribuídos por meio de atributos onevent em instruções innerHTML e outerHTML

Adobe AIR 1.0 e posterior

Ao usar innerHTML ou outerHTML para adicionar elementos ao DOM de um documento, todos os retornos de chamadas de evento atribuídos na instrução, como `onclick` ou `onmouseover`, são ignorados. Nenhum erro de segurança é gerado. Em vez disso, você pode atribuir um atributo `id` aos novos elementos e definir as funções de retorno de chamada do manipulador de eventos, usando o método `addEventListener()`.

Por exemplo, determinado um elemento de destino em um documento, como:

```
<div id="container"></div>
```

Substitua instruções como:

```
document.getElementById('container').innerHTML =
    '<a href="#" onclick="code()">Click Me.</a>';
```

com:

```
document.getElementById('container').innerHTML = '<a href="#" id="smith">Click Me.</a>';
document.getElementById('smith').addEventListener("click", function() { code(); });
```

Carregamento de arquivos JavaScript externos ao diretório de instalação do aplicativo

Adobe AIR 1.0 e posterior

Não é permitido o carregamento de arquivos de script externos à caixa de proteção do aplicativo. Nenhum erro de segurança é gerado. Todos os arquivos de script que são executados na caixa de proteção do aplicativo devem ser instalados no diretório do aplicativo. Para usar scripts externos em uma página, você deve mapear a página para uma caixa de proteção distinta. Consulte “Carregamento de conteúdo do aplicativo em uma caixa de proteção "não aplicativo"” na página 1034.

document.write() e document.writeln()

Adobe AIR 1.0 e posterior

As chamadas para `document.write()` ou `document.writeln()` serão ignoradas após o evento `load` da página ser tratado. Nenhum erro de segurança é gerado. Como alternativa, você pode carregar um novo arquivo ou substituir o corpo do documento usando técnicas de manipulação DOM.

XMLHttpRequests síncronas antes do evento load ou durante um manipulador de eventos load

Adobe AIR 1.0 e posterior

As XMLHttpRequests síncronas iniciadas antes do evento `load` da página ou durante um manipulador de eventos `load` não retornam nenhum conteúdo. As XMLHttpRequests assíncronas podem ser iniciadas, mas não retornam até depois do evento `load`. Após o evento `load` ser tratado, as XMLHttpRequests síncronas se comportam normalmente.

Elementos de script criados dinamicamente

Adobe AIR 1.0 e posterior

Elementos de script criados dinamicamente, como quando criados com o innerHTML ou o método `document.createElement()`, são ignorados.

Acesso às classes API do AIR no JavaScript

Adobe AIR 1.0 e posterior

Além dos elementos padrão e estendidos de kit da Web, o código HTML e JavaScript pode acessar as classes host fornecidas pelo tempo de execução. Essas classes permitem acessar os recursos avançados que o AIR oferece, incluindo:

- Acesso ao sistema de arquivos
- Uso de bancos de dados SQL locais
- Controle de menus de janela e aplicativo
- Acesso a soquetes de rede
- Uso de classes e objetos definidos pelo usuário
- Recursos de som

  Por exemplo, a API de arquivo do AIR inclui uma classe File, contida no pacote flash.filesystem. Você pode criar um objeto File em JavaScript da seguinte forma:

  ```
  var myFile = new window.runtime.flash.filesystem.File();
  ```

  O objeto `runtime` é um objeto JavaScript especial, disponível para conteúdo HTML em execução no AIR na caixa de proteção do aplicativo. Ele permite acessar as classes de tempo de execução do JavaScript. A propriedade `flash` do objeto `runtime` oferece acesso ao pacote flash. Por sua vez, a propriedade `flash.filesystem` do objeto `runtime` oferece acesso ao pacote flash.filesystem (e esse pacote inclui a classe File). Os pacotes são uma maneira de organizar as classes usadas no ActionScript.

  Nota: *A propriedade `runtime` não é adicionada automaticamente aos objetos window de páginas carregadas em um frame ou iframe. No entanto, desde que o documento filho esteja na caixa de proteção do aplicativo, o filho poderá acessar a propriedade `runtime` do pai.*

  Como a estrutura de pacote das classes de tempo de execução exigem que os desenvolvedores digitem longas sequências de código JavaScript para acessar cada classe (como em `window.runtime.flash.desktop.NativeApplication`), o AIR SDK inclui o arquivo AIRAliases.js, que permite acessar as classes de tempo de execução mais facilmente (por exemplo, digitando simplesmente `air.NativeApplication`).

  As classes API do AIR são discutidas em todo este guia. Outras classes da API do Flash Player, que possam ser de interesse dos desenvolvedores HTML, são descritas no *Adobe AIR API Reference for HTML Developers*. ActionScript é a linguagem usada em conteúdo SWF (Flash Player). No entanto, as sintaxes JavaScript e ActionScript são semelhantes. (ambas se baseiam nas versões da linguagem ECMAScript.) Todas as classes incorporadas estão disponíveis em JavaScript (em conteúdo HTML) e ActionScript (em conteúdo SWF).

  Nota: *O código JavaScript não pode usar as classes Dictionary, XML e XMLList, que estão disponíveis no ActionScript.*

Uso do arquivo AIRAliases.js

Adobe AIR 1.0 e posterior

As classes de tempo de execução são organizadas em uma estrutura de pacote, da seguinte forma:

- `window.runtime.flash.desktop.NativeApplication`
- `window.runtime.flash.desktop.ClipboardManager`
- `window.runtime.flash.filesystem.FileStream`
- `window.runtime.flash.data.SQLDatabase`

No AIR SDK está incluído o arquivo AIRAliases.js que oferece definições de "alias" que permitem acessar as classes de tempo de execução com menos digitação. Por exemplo, você pode acessar as classes listadas acima, bastando, para isso, digitar o seguinte:

- `air.NativeApplication`
- `air.Clipboard`
- `air.FileStream`
- `air.SQLDatabase`

A lista é apenas um pequeno subconjunto das classes no arquivo AIRAliases.js. A lista completa de classes e funções em nível de pacote é fornecida no *Adobe AIR API Reference for HTML Developers*.

Além das classes de tempo de execução usadas normalmente, o arquivo AIRAliases.js inclui alias para as funções em nível do pacote usadas com mais frequência. `window.runtime.trace()`, `window.runtime.flash.net.navigateToURL()` e `window.runtime.flash.net.sendToURL()`, com alias `air.trace()`, `air.navigateToURL()` e `air.sendToURL()`.

Para usar o arquivo AIRAliases.js, inclua a seguinte referência de `script` em sua página HTML:

```
<script src="AIRAliases.js"></script>
```

Ajuste o caminho na referência `src`, conforme necessário.

Importante: *Exceto quando observado, o código de exemplo JavaScript nesta documentação pressupõe que você incluiu o arquivo AIRAliases.js em sua página HTML.*

Sobre URLs no AIR

Adobe AIR 1.0 e posterior

No conteúdo HTML em execução no AIR, você pode usar qualquer um dos seguintes esquemas de URL na definição de atributos `src` para `img`, `frame`, `iframe` e tags de `script`, no atributo `href` de uma tag `link` ou qualquer outro local que você possa fornecer uma URL.

esquema de URL	Descrição	Exemplo
arquivo	Um caminho relativo à raiz do sistema de arquivos.	`file:///c:/AIR Test/test.txt`
app	Um caminho relativo à raiz do diretório do aplicativo instalado.	`app:/images`
app-storage	Um caminho relativo ao diretório de armazenamento do aplicativo. Para cada aplicativo instalado, o AIR define um diretório exclusivo de armazenamento do aplicativo, que é um local útil para armazenar dados específicos desse aplicativo.	`app-storage:/settings/prefs.xml`
http	Uma solicitação HTTP padrão.	`http://www.adobe.com`
https	Uma solicitação HTTPS padrão.	`https://secure.example.com`

Para obter mais informações sobre o uso de esquemas de URL no AIR, consulte "Esquemas de URI" na página 840.

Muitas das APIs do AIR, incluindo as classes File, Loader, URLStream e Sound, usam um objeto URLRequest em vez de uma string contendo a URL. O próprio objeto URLRequest é inicializado com uma string, que pode usar qualquer um dos mesmos esquemas de URL. Por exemplo, a seguinte instrução cria um objeto URLRequest que pode ser usado para solicitar a home page da Adobe:

```
var urlReq = new air.URLRequest("http://www.adobe.com/");
```

Para obter informações sobre objetos URLRequest, consulte "Comunicação HTTP" na página 838.

Como tornar objetos ActionScript disponíveis para JavaScript

Adobe AIR 1.0 e posterior

O JavaScript na página HTML carregada pelo objeto HTMLLoader pode chamar as classes, os objetos e as funções definidas no contexto de execução do ActionScript, usando as propriedades `window.runtime`, `window.htmlLoader` e `window.nativeWindow` da página HTML. Você também pode tornar objetos e funções ActionScript disponíveis para código JavaScript, criando referências para eles no contexto de execução do JavaScript.

Um exemplo básico de como acessar objetos JavaScript do ActionScript

Adobe AIR 1.0 e posterior

O seguinte exemplo ilustra como adicionar propriedades que fazem referência a objetos ActionScript ao objeto window global de uma página HTML.

```
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;
}
```

O conteúdo HTML (no arquivo chamado test.html), carregado no objeto HTMLLoader no exemplo anterior, pode acessar a propriedade `foo` e o método `helloFromJS()` definidos no arquivo SWF pai:

```
<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>
```

Ao acessar o contexto JavaScript de um documento que está sendo carregado, você pode usar o evento `htmlDOMInitialize` para criar objetos na sequência de construção da página, cedo o bastante para que qualquer script definido na página possa acessá-los. Se você aguardar o evento `complete`, somente scripts da página executados depois do evento `load` da página poderão acessar os objetos adicionados.

Como tornar as definições de classe disponíveis para JavaScript

Adobe AIR 1.0 e posterior

Para disponibilizar as classes ActionScript do aplicativo em JavaScript, você pode atribuir o conteúdo HTML carregado ao domínio de aplicativo que contém as definições de classe. O domínio de aplicativo do contexto de execução do JavaScript pode ser definido com a propriedade `runtimeApplicationDomain` do objeto HTMLLoader. Para definir o domínio de aplicativo como domínio de aplicativo primário, por exemplo, defina `runtimeApplicationDomain` como `ApplicationDomain.currentDomain`, conforme mostra o código a seguir:

```
html.runtimeApplicationDomain = ApplicationDomain.currentDomain;
```

Depois que a propriedade `runtimeApplicationDomain` for definida, o contexto JavaScript compartilha as definições de classe com o domínio atribuído. Para criar uma ocorrência de classe personalizada em JavaScript, faça referência à definição de classe por meio da propriedade `window.runtime` e use o operador `novo`:

```
var customClassObject = new window.runtime.CustomClass();
```

O conteúdo HTML deve ser de um domínio de segurança compatível. Se o conteúdo HTML for de um domínio de segurança diferente do domínio de aplicativo que você atribuir, em vez disso a página usará um domínio de aplicativo padrão. Por exemplo, se você carregar uma página remota da Internet, não poderá atribuir ApplicationDomain.currentDomain como domínio de aplicativo da página.

Remoção de ouvintes de eventos

Adobe AIR 1.0 e posterior

Quando você adiciona ouvintes de eventos JavaScript a objetos fora da página atual, incluindo objetos de tempo de execução, objetos no conteúdo SWF carregado e até objetos JavaScript em execução em outras páginas, deve sempre remover esses ouvintes de eventos quando a página for descarregada. Do contrário, o ouvinte de evento despacha o evento para uma função do manipulador que não existe mais. Se isso acontecer, você verá a seguinte mensagem de erro: "O aplicativo tentou fazer referência a um objeto JavaScript em uma página HTML que não é mais carregada". A remoção de ouvintes de eventos desnecessários também permite que o AIR recupere a memória associada. Para obter mais informações, consulte “Remoção de ouvintes de eventos nas páginas HTML que navegam” na página 1062.

Acesso a objetos HTML DOM e JavaScript do ActionScript

Adobe AIR 1.0 e posterior

Após o objeto HTMLLoader despachar o evento `complete`, você poderá acessar todos os objetos no HTML DOM (modelo de objeto de documento) da página. Os objetos acessíveis incluem os elementos de exibição (como os objetos `div` e `p` na página), bem como as variáveis e funções JavaScript. O evento `complete` corresponde ao evento `load` da página em JavaScript. Antes que `complete` seja despachado, os elementos DOM, as variáveis e as funções poderão não ter sido analisados nem criados. Se possível, aguarde o evento `complete` antes de acessar o HTML DOM.

Por exemplo, considere a seguinte página HTML:

```
<html>
    <script>
        foo = 333;
        function test() {
            return "OK.";
        }
    </script>
    <body>
        <p id="p1">Hi.</p>
    </body>
</html>
```

Essa página HTML simples define uma variável JavaScript chamada *foo* e uma função JavaScript chamada *test()*. As duas propriedades são propriedades do objeto `window` global da página. Além disso, o objeto `window.document` inclui um elemento chamado P (com a ID *p1*), que você pode acessar usando o método `getElementById()`. Depois que a página for carregada (quando o objeto HTMLLoader despachar o evento `complete`), você poderá acessar cada um desses objetos do ActionScript, conforme mostra o seguinte código ActionScript:

```
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.
}
```

Para acessar o conteúdo de um elemento HTML, use a propriedade `innerHTML`. Por exemplo, o código anterior usa `html.window.document.getElementById("p1").innerHTML` para obter o conteúdo do elemento HTML chamado *p1*.

Você também pode definir propriedades da página HTML do ActionScript. Por exemplo, o seguinte exemplo define o conteúdo do elemento `p1` e o valor da variável JavaScript `foo` na página, usando uma referência ao objeto HTMLLoader que a contém:

```
html.window.document.getElementById("p1").innerHTML = "Goodbye";
html.window.foo = 66;
```

Incorporação de conteúdo SWF em HTML

Adobe AIR 1.0 e posterior

Você pode incorporar o conteúdo SWF em um conteúdo HTML do aplicativo do AIR da mesma forma que em um navegador. Incorpore o conteúdo SWF usando a tag `object`, a tag `embed` ou as duas.

Nota: *Uma prática comum de desenvolvimento é usar tanto uma tag `object` quanto uma tag `embed` para exibir conteúdo SWF em uma página HTML. Essa prática não tem nenhum benefício no AIR. Você pode usar a própria tag `object` padrão W3C no conteúdo para ser exibido no AIR. Ao mesmo tempo, você pode continuar usando as tags `object` e `embed` juntas, se necessário, para conteúdo HTML exibido também no navegador.*

Se você habilitou a transparência no objeto NativeWindow exibindo o conteúdo HTML e SWF, o AIR não exibe o conteúdo SWF quando o modo janela (`wmode`) utilizado para integrar o conteúdo estiver definido em: `window`. Para exibir o conteúdo do SWF em uma página HTML de uma janela transparente, configure o parâmetro `wmode` para `opaque` ou `transparent`. `window` é o valor padrão para `wmode`, de forma que o conteúdo pode não ser exibido se você não especificar o valor.

O seguinte exemplo ilustra o uso da tag `object` HTML para exibir um arquivo SWF no conteúdo HTML. O parâmetro `wmode` está definido para `opaque` de forma que o conteúdo é exibido, mesmo quando o objeto base NativeWindow for transparente. O arquivo SWF é carregado do diretório do aplicativo, mas você pode usar qualquer um dos esquemas de URL suportados pelo AIR. (O local do qual o arquivo SWF é carregado determina a caixa de proteção de segurança em que o AIR coloca o conteúdo.)

```
<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>
```

Você também pode usar um script para carregar conteúdo dinamicamente. O seguinte exemplo cria um nó `object` para exibir o arquivo SWF especificado no parâmetro `urlString`. O exemplo adiciona o nó como filho do elemento de página com a ID especificada pelo parâmetro `elementID`:

```
<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>
```

O conteúdo SWF não é exibido se o objeto HTMLLoader for dimensionado ou girado, ou se a propriedade `alpha` estiver definida para um valor diferente de 1.0. Antes do AIR 1.5.2, o conteúdo SWF não era exibido em uma janela transparente, não importando como o valor `wmode` estava definido.

Nota: *Quando um objeto SWF incorporado tentar carregar um ativo externo como um arquivo de vídeo, o conteúdo do SWF poderá não ser renderizado adequadamente se não for fornecido um caminho absoluto para o vídeo no arquivo HTML. Contudo, um objeto SWF incorporado pode carregar um arquivo de imagem externo usando um caminho relativo.*

O exemplo a seguir demonstra como os ativos externos podem ser carregados por meio de um objeto SWF incorporado num conteúdo HTML:

```
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");

}
```

No exemplo de ActionScript a seguir, o caminho da imagem passado pelo arquivo HTML é lido, e a imagem é carregada no palco:

```
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;
                  }
            }
      }
}
```

Uso de bibliotecas do ActionScript em uma página HTML

Adobe AIR 1.0 e posterior

O AIR estende o elemento de script HTML para que a página possa importar classes ActionScript em um arquivo SWF compilado. Por exemplo, para importar uma biblioteca chamada *myClasses.swf*, localizada no subdiretório `lib` da pasta raiz do aplicativo, inclua a seguinte tag de script no arquivo HTML:

```
<script src="lib/myClasses.swf" type="application/x-shockwave-flash"></script>
```

Importante: *O atributo de tipo deve ser* `type="application/x-shockwave-flash"` *para que a biblioteca seja carregada corretamente.*

Se o conteúdo SWF for compilado como Flash Player 10 ou AIR 1.5 SWF, você deverá definir o namespace do descritor do aplicativo como namespace do AIR 1.5.

O diretório `lib` e o arquivo `myClasses.swf` também deve ser incluído quando o arquivo AIR for empacotado.

Acesse as classes importadas por meio da propriedade `runtime` do objeto da janela JavaScript:

```
var libraryObject = new window.runtime.LibraryClass();
```

Se as classes no arquivo SWF estiverem organizadas em pacotes, você também deverá incluir o nome do pacote. Por exemplo, se a definição LibraryClass estivesse em um pacote chamado *utilities*, você deveria criar uma ocorrência da classe com a seguinte instrução:

```
var libraryObject = new window.runtime.utilities.LibraryClass();
```

Nota: *Para compilar uma biblioteca SWF do ActionScript para usar como parte da página HTML no AIR, use o compilador* `acompc`*. O utilitário acompc é parte do SDK do Flex e está descrito na _brDocumentação do SDK Flex.*

Acesso aos objetos HTML DOM e JavaScript de um arquivo ActionScript importado

Adobe AIR 1.0 e posterior

Para acessar objetos em uma página HTML do ActionScript em um arquivo SWF importado na página usando a tag `<script>`, passe uma referência para um objeto JavaScript, como `window` ou `document`, em uma função definida no código do ActionScript. Use a referência na função para acessar o objeto JavaScript (ou outros objetos acessíveis por meio da referência passada).

Por exemplo, considere a seguinte página HTML:

```
<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>
```

Essa página HTML simples tem uma variável JavaScript chamada *num* e uma função JavaScript chamada *getStatus()*. As duas propriedades são propriedades do objeto `window` da página. Além disso, o objeto `window.document` inclui um elemento chamado P (com a ID *p1*).

A página carrega o arquivo ActionScript, "ASLibrary.swf," que contém a classe ASClass. A ASClass define a função chamada `accessDOM()` que simplesmente rastreia os valores desses objetos JavaScript. O método `accessDOM()` considera o objeto de janela JavaScript como um argumento. Ao usar essa referência de janela, ele pode acessar outros objetos na página, incluindo variáveis, funções e elementos DOM, conforme ilustrado na definição a seguir:

```
public class ASClass{
    public function accessDOM(window:*):void {
        trace(window.num); // 254
        trace(window.document.getElementById("p1").innerHTML); // Body text..
        trace(window.getStatus()); // OK.
    }
}
```

Você pode obter e definir propriedades da página HTML de uma classe ActionScript importada. Por exemplo, a seguinte função define o conteúdo do elemento `p1` na página e define o valor da variável de JavaScript `foo` na página:

```
public function modifyDOM(window:*):void {
    window.document.getElementById("p1").innerHTML = "Bye";
    window.foo = 66;
```

Conversão de objetos Date e RegExp

Adobe AIR 1.0 e posterior

As linguagens JavaScript e ActionScript definem as classes Date e RegExp, mas os objetos desses tipos não são convertidos automaticamente entre os dois contextos de execução. Você deve converter os objetos Date e RegExp para o tipo equivalente antes de usá-los para definir parâmetros de propriedades ou funções no contexto de execução alternativo.

Por exemplo, o seguinte código ActionScript converte o objeto Date de JavaScript chamado `jsDate` em um objeto Date do ActionScript:

```
var asDate:Date = new Date(jsDate.getMilliseconds());
```

Por exemplo, o seguinte código ActionScript converte o objeto RegExp de JavaScript chamado `jsRegExp` em um objeto RegExp do ActionScript:

```
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);
```

Manipulação de folha de estilos HTML do ActionScript

Adobe AIR 1.0 e posterior

Depois que o objeto HTMLLoader tiver despachado o evento `complete`, você poderá examinar e manipular estilos CSS na página.

Por exemplo, considere o seguinte documento HTML simples:

```
<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>
```

Depois que o objeto HTMLLoader carregar esse conteúdo, você poderá manipular os estilos CSS da página, através da matriz `cssRules` da matriz `window.document.styleSheets`, como mostrado a seguir:

```
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";
}
```

Esse código ajusta os estilos CSS para que o documento HTML resultante tenha a seguinte aparência:

Style 1A

Style 1B

Style 2

Lembre-se de que esse código pode adicionar estilos à página após o objeto HTMLLoader despachar o evento `complete`.

Conteúdo cross-scripting em caixas de proteção de segurança distintas

Adobe AIR 1.0 e posterior

O modelo de segurança de tempo de execução isola o código de origens distintas. Cruzando scripts de conteúdo em caixas de proteção de segurança distintas, você pode permitir que o conteúdo de uma caixa de proteção de segurança acesse as propriedades e os métodos selecionados em outra caixa de proteção.

Caixas de proteção de segurança do AIR e código JavaScript

Adobe AIR 1.0 e posterior

O AIR aplica a política de mesma origem, que impede a interação de código de um domínio com o conteúdo de outro domínio. Todos os arquivos são colocados em uma caixa de proteção com base nas respectivas origens. Normalmente, o conteúdo na caixa de proteção do aplicativo não pode violar o princípio de mesma origem e o conteúdo cross-scripting, carregado de fora do diretório de instalação do aplicativo. No entanto, o AIR oferece algumas técnicas que permitem o cruzamento de scripts em conteúdo "não aplicativo".

Uma das técnicas usa frames ou iframes para mapear conteúdo de aplicativo em uma caixa de proteção de segurança distinta. Qualquer página carregada da área com caixa de proteção do aplicativo se comporta como se tivesse sido carregada do domínio remoto. Por exemplo, mapeando o conteúdo do aplicativo para o domínio *example.com*, esse conteúdo pode fazer o cruzamento cross-scripting das páginas carregadas desse domínio.

Como essa técnica coloca o conteúdo do aplicativo em uma caixa de proteção distinta, o código nesse conteúdo também não estará mais sujeito às restrições na execução de código em strings avaliadas. Você pode usar essa técnica de mapeamento de caixa de proteção para para atenuar essas restrições, mesmo quando não for necessário fazer o cruzamento de scripts de conteúdo remoto. Mapear conteúdo dessa maneira pode ser muito útil ao trabalhar com uma das várias estruturas JavaScript ou com código existente que dependa das strings de avaliação. No entanto, você deve considerar e se proteger contra o risco adicional de que algum conteúdo não confiável possa ser injetado e executado quando o conteúdo for executado fora da caixa de proteção do aplicativo.

Ao mesmo tempo, o conteúdo de aplicativo mapeado para uma outra caixa de proteção perde o acesso às APIs do AIR, portanto, a técnica de mapeamento de caixa de proteção não pode ser usada para expor a funcionalidade do AIR ao código executado fora da caixa de proteção do aplicativo.

Outra técnica de cruzamento de scripts permite criar uma interface chamada *ponte de caixa de proteção* entre o conteúdo de uma caixa de proteção "não aplicativo" e seu documento pai na caixa de proteção do aplicativo. A ponte permite que o conteúdo filho acesse as propriedades e os métodos definidos pelo pai e o pai acesse as propriedades e os métodos definidos pelo filho, ou ambos.

Por fim, você também pode executar XMLHttpRequests entre domínios da caixa de proteção do aplicativo e, opcionalmente, de outras caixas de proteção.

Para obter mais informações, consulte “Elementos HTML frame e iframe” na página 1005, “Segurança HTML no Adobe AIR” na página 1121 e “O objeto XMLHttpRequest” na página 999.

Carregamento de conteúdo do aplicativo em uma caixa de proteção "não aplicativo"

Adobe AIR 1.0 e posterior

Para permitir que o conteúdo do aplicativo faça cruzamento de script de conteúdo carregado de fora do diretório de instalação do aplicativo, você pode usar os elementos `frame` ou `iframe` para carregar conteúdo do aplicativo na mesma caixa de proteção de segurança do conteúdo externo. Se você não precisa fazer cruzamento de script de conteúdo remoto, mas ainda assim deseja carregar uma página do seu aplicativo fora da respectiva caixa de proteção, pode usar a mesma técnica, especificando `http://localhost/` ou algum outro valor inócuo, como o domínio de origem.

O AIR adiciona os novos atributos `sandboxRoot` e `documentRoot` ao elemento frame que permite especificar se o arquivo do aplicativo carregado no frame deve ser mapeado para uma caixa de proteção "não aplicativo". Arquivos que estão sendo resolvidos em um caminho abaixo da URL `sandboxRoot` são carregados, em vez do diretório `documentRoot`. Para fins de segurança, o conteúdo do aplicativo carregado dessa maneira é tratado como se na verdade tivesse sido carregado da URL `sandboxRoot`.

A propriedade `sandboxRoot` especifica a URL que deve ser usada para determinar a caixa de proteção e o domínio em que o conteúdo do quadro deve ser colocado. Os esquemas de URL `file:`, `http:` ou `https:` devem ser usados. Se você especificar uma URL relativa, o conteúdo permanecerá na caixa de proteção do aplicativo.

A propriedade `documentRoot` especifica o diretório do qual o conteúdo do quadro deve ser carregado. Os esquemas de URL `file:`, `app:` ou `app-storage:` devem ser usados.

O exemplo a seguir mapeia o conteúdo instalado no subdiretório `sandbox` do aplicativo a ser executado na caixa de proteção remota e o domínio `www.example.com`:

```
 <iframe
    src="http://www.example.com/local/ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

A página `ui.html` pode carregar um arquivo javascript da pasta `sandbox` local, usando a seguinte tag de script:

```
<script src="http://www.example.com/local/ui.js"></script>
```

Ele também pode carregar conteúdo de um diretório no servidor remoto usando uma tag de script como a que segue:

```
<script src="http://www.example.com/remote/remote.js"></script>
```

A URL `sandboxRoot` irá mascarar todo conteúdo na mesma URL do servidor remoto. No exemplo anterior, você não podia acessar nenhum conteúdo remoto em `www.example.com/local/` (ou qualquer um de seus subdiretórios), pois o AIR remapeia a solicitação para o diretório local do aplicativo. As solicitações são remapeadas, sejam elas derivadas de navegação de página, de uma XMLHttpRequest ou de qualquer outro meio de carregamento de conteúdo.

Configuração de interface de ponte de caixa de proteção

Adobe AIR 1.0 e posterior

Você pode usar uma ponte de caixa de proteção quando o conteúdo da caixa de proteção do aplicativo tiver que acessar propriedades ou métodos definidos pelo conteúdo em uma caixa de proteção "não aplicativo" , ou quando o conteúdo "não aplicativo" tiver que acessar propriedades e métodos definidos pelo conteúdo na caixa de proteção do aplicativo. Crie uma ponte com as propriedades `childSandboxBridge` e `parentSandboxBridge` do objeto `window` de algum documento filho.

Estabelecimento de uma ponte de caixa de proteção filha

Adobe AIR 1.0 e posterior

A propriedade `childSandboxBridge` permite que o documento filho exponha uma interface para o conteúdo do documento pai. Para expor uma interface, você define a propriedade `childSandbox` como função ou objeto no documento filho. Em seguida, você pode acessar o objeto ou função do conteúdo no documento pai. O exemplo a seguir mostra como um script que está sendo executado em um documento filho pode expor um objeto contendo uma função e uma propriedade para o respectivo pai:

```
var interface = {};
interface.calculatePrice = function(){
    return ".45 cents";
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Se esse filho foi carregado em um iframe com id "filho" atribuída, você poderá acessar a interface do conteúdo pai, lendo a propriedade `childSandboxBridge` do quadro:

```
var childInterface = document.getElementById("child").contentWindow.childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces ".45 cents"
air.trace(childInterface.storeID)); //traces "abc"
```

Estabelecimento de uma ponte de caixa de proteção pai

Adobe AIR 1.0 e posterior

A propriedade `parentSandboxBridge` permite que o documento pai exponha uma interface para o conteúdo do documento filho. Para expor uma interface, o documento pai define a propriedade `parentSandbox` do documento filho como uma função ou objeto definido no documento pai. Em seguida, você pode acessar o objeto ou função do conteúdo no filho. O exemplo a seguir mostra como um script que está sendo executado em um quadro pai pode expor um objeto contendo uma função para um documento filho:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").contentWindow.parentSandboxBridge = interface;
```

Ao usar essa interface, o conteúdo do quadro filho poderá salvar texto em um arquivo chamado `save.txt`, mas não terá nenhum outro acesso ao sistema de arquivos. O conteúdo filho poderá chamar a função save da seguinte maneira:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

O conteúdo do aplicativo deverá expor a interface mais estreita possível para as outras caixas de proteção. O conteúdo "não aplicativo" deve ser considerado não confiável inerentemente, já que ele pode estar sujeito à injeção de código acidental ou mal-intencionado. Você deve colocar as proteções apropriadas no local para impedir o uso inadequado da interface exposta através da ponte da caixa de proteção pai.

Acesso à ponte de caixa de proteção pai durante o carregamento de página

Adobe AIR 1.0 e posterior

Para que o script de um documento filho acesse uma ponte de caixa de proteção pai, a ponte deve ser configurada antes que o script seja executado. Os objetos window, frame e iframe despacham o evento `dominitialize` quando uma nova página DOM tiver sido criada, mas antes que qualquer script tenha sido analisado ou que elementos DOM tenham sido adicionados. Você pode usar o evento `dominitialize` para estabelecer a ponte na sequência de construção de página, cedo o bastante para que todos os scripts no documento filho possam acessá-la.

O exemplo a seguir ilustra como criar uma ponte de caixa de proteção pai em resposta ao evento `dominitialize` despachado do quadro filho:

```
<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>
```

O seguinte documento `child.html` ilustra como o conteúdo filho pode acessar a ponte de caixa de proteção pai:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Para ouvir o evento `dominitialize` em uma janela filha, em vez de um quadro, você deve adicionar o ouvinte ao novo objeto window filho criado pela função `window.open()`:

```
var childWindow = window.open();
childWindow.addEventListener("dominitialize", engageBridge());
childWindow.document.location = "http://www.example.com/air/child.html";
```

Nesse caso, não há como mapear o conteúdo do aplicativo para uma caixa de proteção "não aplicativo". Essa técnica só é útil quando `child.html` é carregado de fora do diretório do aplicativo. Você pode ainda mapear conteúdo do aplicativo na janela para uma caixa de proteção "não aplicativo", mas primeiramente, é preciso carregar uma página intermediária que use ela mesma quadros para carregar o documento filho e mapeá-lo para a caixa de proteção desejada.

Se você usar a função `createRootWindow()` da classe HTMLLoader para criar uma janela, a nova janela não será filha do documento do qual `createRootWindow()` será chamado. Portanto, você não pode criar uma ponte de caixa de proteção da janela que faz a chamada para um conteúdo "não aplicativo" carregado na nova janela. Em vez disso, você deve carregar uma página intermediária na nova janela que use ela mesma quadros para carregar o documento filho. Em seguida, você pode estabelecer a ponte do documento pai da nova janela para o documento filho carregado no quadro.

Capítulo 60: Script no contêiner HTML do AIR

Adobe AIR 1.0 e posterior

A classe HTMLLoader serve como contêiner do conteúdo HTML no Adobe® AIR®. A classe oferece várias propriedades e métodos, herdados da classe Sprite, para controlar o comportamento e a aparência do objeto na lista de exibição do ActionScript® 3.0. Além disso, a classe define propriedades e métodos de tarefas, como carregamento e interação com conteúdo HTML e gerenciamento de histórico.

A classe HTMLHost define um conjunto de comportamentos de um HTMLLoader. Quando você cria um objeto HTMLLoader, nenhuma implementação HTMLHost é fornecida. Portanto, quando o conteúdo HTML aciona um dos comportamentos padrão, como alteração de localização da janela ou do título da janela, não acontece nada. Você pode estender a classe HTMLHost para definir os comportamentos desejados para o aplicativo.

Uma implementação padrão do HTMLHost é fornecida para janelas HTML criadas pelo AIR. Você pode atribuir a implementação HTMLHost padrão a outro objeto HTMLLoader definindo a propriedade `htmlHost` do objeto, usando um novo objeto HTMLHost criado com o parâmetro `defaultBehavior` definido como `true`.

Nota: *Na estrutura do Adobe® Flex™, o objeto HTMLLoader é delimitado pelo componente mx:HTML. Use o componente HTML quando usar o Flex.*

Exibição de propriedades de objetos HTMLLoader

Adobe AIR 1.0 e posterior

O objeto HTMLLoader herda as propriedades de exibição da classe Sprite do Adobe® Flash® Player. Você pode redimensionar, mover, ocultar e alterar a cor do plano de fundo, por exemplo. Ou, pode aplicar efeitos avançadas, como filtros, máscaras, dimensionamento e rotação. Ao aplicar efeitos, considere o impacto sobre a legibilidade. O conteúdo SWF e PDF carregado em uma página HTML não pode ser exibido quando alguns efeitos são aplicados.

As janelas HTML contêm um objeto HTMLLoader que processa conteúdo HTML. Esse objeto fica restrito dentro da área da janela, portanto, alterar dimensões, posição, rotação ou fato de dimensionamento nem sempre produz os resultados desejados.

Propriedades básicas de exibição

Adobe AIR 1.0 e posterior

As propriedades básicas de exibição do HTMLLoader permitem posicionar o controle no respectivo objeto de exibição pai para definir o tamanho e mostrar ou ocultar o controle. Você não deve alterar essas propriedades do objeto HTMLLoader de uma janela HTML.

As propriedades básicas incluem:

Propriedade	Observações
`x`, `y`	Posiciona o objeto em seu contêiner pai.
`width`, `height`	Altera as dimensões da área de exibição.
`visible`	Controla a visibilidade do objeto e todo seu conteúdo.

Fora da janela HTML, as propriedades `width` e `height` do objeto HTMLLoader assumem o padrão 0. Você deve definir a largura e a altura para que o conteúdo HTML carregado possa ser visualizado. O conteúdo HTML é desenhado no tamanho do HTMLLoader, disposto de acordo com as propriedades HTML e CSS do conteúdo. Alterar o tamanho do HTMLLoader redireciona o conteúdo.

Ao carregar conteúdo em um novo objeto HTMLLoader (com `width` ainda definido como 0), pode ser tentador definir `width` e `height` de exibição do HTMLLoader usando as propriedades `contentWidth` e `contentHeight`. Essa técnica funciona em páginas com largura mínima razoável quando disposta de acordo com as regras de fluxo de HTML e CSS. No entanto, algumas páginas resultam em um layout longo e estreito na ausência de uma largura razoável fornecida pelo HTMLLoader.

Nota: *Quando você altera a largura e a altura do objeto HTMLLoader, os valores scaleX e scaleY não são alterados, como ocorreria com a maioria dos demais tipos de objetos de exibição.*

Transparência de conteúdo HTMLLoader

Adobe AIR 1.0 e posterior

A propriedade `paintsDefaultBackground` do objeto HTMLLoader, que é `true` por padrão, determina se o objeto HTMLLoader desenha um plano de fundo opaco. Quando `paintsDefaultBackground` for `false`, o plano de fundo ficará claro. O contêiner do objeto de exibição ou outros objetos de exibição abaixo do objeto HTMLLoader ficam visíveis por trás dos elementos de primeiro plano do conteúdo HTML.

Se o elemento body ou qualquer outro elemento do documento HTML especificar uma cor de plano de fundo (usando `style="background-color:gray"`, por exemplo), o plano de fundo dessa parte do HTML ficará opaco e será processado com a cor de plano de fundo especificada. Se você definir a propriedade `opaqueBackground` do objeto HTMLLoader e `paintsDefaultBackground` for `false`, a cor definida para `opaqueBackground` ficará visível.

Nota: *Você pode usar um gráfico com formato PNG transparente para dar ao elemento no documento HTML um plano de fundo de mesclagem de alfa. Não há suporte para a configuração de estilo de opacidade de elemento HTML.*

Dimensionamento de conteúdo HTMLLoader

Adobe AIR 1.0 e posterior

Evite o dimensionamento do objeto HTMLLoader além do fator de dimensionamento de 1.0. O texto no conteúdo HTMLLoader é processado em uma resolução específica e aparecerá pixelizado se o objeto HTMLLoader for dimensionado. Para evitar que o HTMLLoader, bem como o respectivo conteúdo, sejam dimensionados quando a janela for redimensionada, defina a propriedade `scaleMode` do Palco como `StageScaleMode.NO_SCALE`.

Considerações ao carregar conteúdo SWF ou PDF em uma página HTML

Adobe AIR 1.0 e posterior

O conteúdo SWF e PDF carregado em um objeto HTMLLoader desaparece nas seguintes condições:

- Se você dimensionar o objeto HTMLLoader em um outro fator que não 1.0.
- Se você definir a propriedade alfa do objeto HTMLLoader em um outro valor que não 1.0.
- Se você girar o conteúdo HTMLLoader.

O conteúdo reaparecerá se você remover a configuração da propriedade ofensiva e os filtros ativos.

Além disso, o tempo de execução não pode exibir o conteúdo de arquivos PDF em janelas transparentes. O tempo de execução somente pode exibir conteúdo SWF integrado em uma página HTML quando o parâmetro `wmode` do objeto ou da tag integrada estiver definido em `opaque` ou `transparent`. Como o valor padrão de `wmode` é `window`, o conteúdo SWF não é exibido em janelas transparentes a não ser que você especifique expressamente o parâmetro wmode.

Nota: *Antes da versão 1.5.2 do AIR, o SWF integrado no HTML não podia ser exibido, não importando como o valor wmode era utilizado.*

Para obter mais informações sobre o carregamento desses tipos de mídia no HTMLLoader, consulte "Incorporação de conteúdo SWF em HTML" na página 1026 e "Inclusão de conteúdo em PDF no AIR" na página 568.

Propriedades avançadas de exibição

Adobe AIR 1.0 e posterior

A classe HTMLLoader herda vários métodos que podem ser usados em efeitos especiais. Em geral, esses efeitos têm limitações quando usados com a exibição HTMLLoader, mas podem ser úteis em transições ou outros efeitos temporários. Por exemplo, se você exibe uma janela de diálogo para coletar entradas de usuário, é possível desfocar a exibição da janela principal até que o usuário feche a caixa de diálogo. Da mesma forma, você pode fazer a exibição desaparecer gradualmente ao fechar uma janela.

As propriedades avançadas de exibição incluem:

Propriedade	Limitações
`alpha`	Pode reduzir a legibilidade do conteúdo HTML.
`filters`	Em uma Janela HTML, os efeitos exteriores são recortados pela borda da janela.
`graphics`	As formas desenhadas com comandos gráficos aparecem abaixo do conteúdo HTML, incluindo o plano de fundo padrão. A propriedade paintsDefaultBackground deve ser false para que as formas do desenho estejam visíveis.
`opaqueBackground`	Não altera a cor do plano de fundo padrão. A propriedade paintsDefaultBackground deve ser false para que essa camada de cor esteja visível.

Propriedade	Limitações
`rotation`	Os cantos da área retangular do HTMLLoader podem ser recortados pela borda da janela. O conteúdo SWF e PDF carregado no conteúdo HTML não é exibido.
`scaleX` e `scaleY`	A exibição processada pode aparecer pixelizada em fatores de dimensionamento maiores que 1. O conteúdo SWF e PDF carregado no conteúdo HTML não é exibido.
`transform`	Pode reduzir a legibilidade do conteúdo HTML. A exibição HTML pode ser recortada pela borda da janela. O conteúdo SWF e PDF carregado no conteúdo HTML não será exibido se a transformação envolver rotação, dimensionamento ou inclinação.

O exemplo a seguir ilustra como definir a matriz `filters` para desfocar a exibição HTML inteira:

```
 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;
```

Rolagem de conteúdo HTML

Adobe AIR 1.0 e posterior

A classe HTMLLoader inclui as seguintes propriedades que permitem controlar a rolagem de conteúdo HTML:

Propriedade	Descrição
`contentHeight`	A altura, em pixels, do conteúdo HTML.
contentWidth	A largura, em pixels, do conteúdo HTML.
scrollH	A posição da barra de rolagem horizontal do conteúdo HTML no objeto HTMLLoader.
scrollV	A posição da barra de rolagem vertical do conteúdo HTML no objeto HTMLLoader.

O código a seguir define a propriedade `scrollV`, de modo que o conteúdo HTML seja rolado para a parte inferior da página:

```
 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;
}
```

O HTMLLoader não inclui barras de rolagem horizontal e vertical. Você pode implementar barras de rolagem no ActionScript ou usando um componente Flex. O componente HTML do Flex inclui automaticamente barras de rolagem para conteúdo HTML. Você também pode usar o método `HTMLLoader.createRootWindow()` para criar uma janela contendo um objeto HTMLLoader com barras de rolagem (consulte "Criação de janelas com conteúdo HTML de rolagem" na página 1054).

Acesso à lista de histórico de HTML

Adobe AIR 1.0 e posterior

À medida que novas páginas são carregadas no objeto HTMLLoader, o tempo de execução mantém uma lista de histórico do objeto. A lista de histórico corresponde ao objeto `window.history` na página HTML. A classe HTMLLoader inclui as seguintes propriedades e métodos que permitem trabalhar com a lista de histórico de HTML:

Membro de classe	Descrição
`historyLength`	O comprimento geral da lista de histórico, incluindo entradas dianteiras e traseiras.
`historyPosition`	A posição atual na lista de histórico. Os itens de histórico antes dessa posição representam navegação "para trás" e itens após essa posição representam navegação "para frente".
`getHistoryAt()`	Retorna o objeto URLRequest correspondente à entrada de histórico na posição especificada na lista de histórico.
`historyBack()`	Navega para trás na lista de histórico, se possível.
`historyForward()`	Navega para frente na lista de histórico, se possível.
`historyGo()`	Navega o número indicado de etapas no histórico do navegador. Navega para frente, se for positivo, e para trás, se for negativo. Navegar para zero recarrega a página. Especificar uma posição além do final, navega para o final da lista.

Itens na lista de histórico são armazenados como objetos do tipo HTMLHistoryItem. A classe HTMLHistoryItem possui as propriedades a seguir:

Propriedade	Descrição
`isPost`	Defina como `true` se a página HTML incluir dados POST.
`originalUrl`	A URL original da página HTML antes de qualquer redirecionamento.
`título`	O título da página HTML.
`url`	A URL da página HTML.

Definição do agente do usuário usado ao carregar conteúdo HTML

Adobe AIR 1.0 e posterior

A classe HTMLLoader tem a propriedade `userAgent`, que permite definir a sequência de agente do usuário usada pelo HTMLLoader. Defina a propriedade `userAgent` do objeto HTMLLoader antes de chamar o método `load()`. Se você definir esta propriedade na ocorrência HTMLLoader, a propriedade `userAgent` do URLRequest passada para o método `load()` *não* será usada.

Você pode definir a string de agente do usuário padrão usada por todos os objetos HTMLLoader em um domínio de aplicativo, configurando a propriedade `URLRequestDefaults.userAgent`. As propriedades URLRequestDefaults estáticas aplicam-se como padrão para todos os objetos URLRequest e não somente URLRequests usadas no método `load()` de objetos HTMLLoader. Configurar a propriedade `userAgent` do HTMLLoader substitui a configuração `URLRequestDefaults.userAgent` padrão.

Se você não definir um valor de agente do usuário para a propriedade `userAgent` do objeto HTMLLoader ou para `URLRequestDefaults.userAgent`, o valor de agente do usuário AIR padrão será usado. Esse valor padrão varia em função do sistema operacional do tempo de execução (tal como Mac OS ou Windows), do idioma do tempo de execução e da versão, como nos dois exemplos a seguir:

- `"Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/420+ (KHTML, como Gecko) AdobeAIR/1.0"`
- `"Mozilla/5.0 (Windows; U; en) AppleWebKit/420+ (KHTML, como Gecko) AdobeAIR/1.0"`

Configuração de codificação de caractere para uso de conteúdo HTML.

Adobe AIR 1.0 e posterior

A página HTML pode especificar a codificação de caractere por ela usado, incluindo a tag `meta`, como a seguir:

```
meta http-equiv="content-type" content="text/html" charset="ISO-8859-1";
```

Substitua a configuração da página para assegurar que a codificação de caractere específica seja usada, configurando a propriedade `textEncodingOverride` do objeto HTMLLoader:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingOverride = "ISO-8859-1";
```

Especifique a codificação de caractere do conteúdo HTMLLoader que deverá ser usada quando a página HTML não especificar uma configuração com a propriedade `textEncodingFallback` do objeto HTMLLoader:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingFallback = "ISO-8859-1";
```

A propriedade `textEncodingOverride` substitui a configuração da página HTML. E a propriedade `textEncodingOverride` e a configuração da página HTML substituem a propriedade `textEncodingFallback`.

Defina a propriedade `textEncodingOverride` ou a propriedade `textEncodingFallback` antes de carregar conteúdo HTML.

Definição de interfaces do usuário como navegadores para conteúdo HTML

Adobe AIR 1.0 e posterior

O JavaScript oferece diversas APIs para controlar a janela que exibe o conteúdo HTML. No AIR, essas APIs podem ser sobrescritas com a implementação de uma classe personalizada HTMLHost.

Sobre estender a classe HTMLHost

Adobe AIR 1.0 e posterior

Se, por exemplo, o aplicativo apresentar vários objetos HTMLLoader em uma interface com abas, talvez você deseje que as alterações feitas pelas páginas HTML carregadas alterem o rótulo da aba e não o título da janela principal. Da mesma forma, o código pode responder a uma chamada `window.moveTo()` reposicionando o objeto HTMLLoader no respectivo contêiner do objeto de exibição pai, movendo a janela que contém o objeto HTMLLoader, não fazendo exatamente nada ou fazendo alguma outra coisa por completo.

A classe HTMLHost do AIR controla as seguintes propriedades e métodos JavaScript:

- `window.status`
- `window.document.title`
- `window.location`
- `window.blur()`
- `window.close()`
- `window.focus()`
- `window.moveBy()`
- `window.moveTo()`
- `window.open()`
- `window.resizeBy()`
- `window.resizeTo()`

Quando você cria um objeto HTMLLoader usando `new HTMLLoader()`, as propriedades ou métodos JavaScript listados não são ativados. A classe HTMLHost oferece uma implementação padrão como navegador dessas APIs JavaScript. Você também pode estender a classe HTMLHost para personalizar o comportamento. Para criar um objeto HTMLHost que ofereça suporte ao comportamento padrão, defina o parâmetro `defaultBehaviors` como true no construtor HTMLHost:

```
var defaultHost:HTMLHost = new HTMLHost(true);
```

Quando você cria uma janela HTML no AIR com o método `createRootWindow()` da classe HTMLLoader, uma ocorrência HTMLHost com suporte aos comportamentos padrão é atribuída automaticamente. Você pode alterar o comportamento do objeto host atribuindo uma implementação HTMLHost diferente à propriedade `htmlHost` do HTMLLoader ou, pode atribuir `null` para desativar os recursos completamente.

Nota: *O AIR atribui um objeto HTMLHost padrão à janela inicial criada para o aplicativo do AIR baseado em HTML e qualquer janela criada pela implementação padrão do método `window.open()` de JavaScript.*

Exemplo: Extensão da classe HTMLHost

Adobe AIR 1.0 e posterior

O exemplo a seguir mostra como personalizar a maneira como o objeto HTMLLoader afeta a interface do usuário, estendendo a classe HTMLHost:

Exemplo do Flex:

1. Crie uma classe que estenda a classe HTMLHost (uma subclasse).
2. Substitua os métodos da nova classe para tratar alterações nas configurações relacionadas à interface do usuário. Por exemplo, a classe a seguir, CustomHost, define comportamentos de chamadas para`window.open()` e altera para `window.document.title`. As chamadas para `window.open()` abrem a página HTML em uma nova janela e alterações em `window.document.title` (incluindo a configuração do elemento `<title>` da página HTML) definem o título dessa janela.

```
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 No código que contém o HTMLLoader (não no código da nova subclasse de HTMLHost), crie um objeto da nova classe. Atribua o novo objeto à propriedade `htmlHost` do HTMLLoader. O seguinte código Flex usa a classe CustomHost definida na etapa anterior:

```
<?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>
```

Para testar o código descrito aqui, inclua um arquivo HTML com o seguinte conteúdo no diretório do aplicativo:

```
<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>
```

Exemplo do Flash Professional:

1 Crie um arquivo Flash para o AIR. Defina a classe do documento como CustomHostExample e, em seguida, salve o arquivo como CustomHostExample.fla.

2 Crie um arquivo ActionScript chamado CustomHost.as contendo uma classe que estenda a classe HTMLHost (uma subclasse). Essa classe substitui certos métodos da nova classe para tratar alterações nas configurações relacionadas à interface do usuário. Por exemplo, a classe a seguir, CustomHost, define comportamentos de chamadas para`window.open()` e altera para `window.document.title`. As chamadas para o método `window.open()` abrem a página HTML em uma nova janela e alterações na propriedade `window.document.title` (incluindo a configuração do elemento `<title>` da página HTML) definem o título dessa janela.

```
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 Crie outro arquivo ActionScript chamado CustomHostExample.as para conter a classe de documento do aplicativo. Essa classe cria um objeto HTMLLoader e define a respectiva propriedade host como uma ocorrência da classe CustomHost definida na etapa anterior:

```
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;
        }
    }
}
```

Para testar o código descrito aqui, inclua um arquivo HTML com o seguinte conteúdo no diretório do aplicativo:

```
<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>
```

Tratamento de alterações na propriedade window.location

Adobe AIR 1.0 e posterior

Substitua o método `locationChange()` para tratar alterações da URL da página HTML. O método `locationChange()` é chamado quando o JavaScript em uma página altera o valor de `window.location`. O exemplo a seguir simplesmente carrega a URL solicitada:

```
override public function updateLocation(locationURL:String):void
{
    htmlLoader.load(new URLRequest(locationURL));
}
```

Nota: *Você pode usar a propriedade htmlLoader do objeto HTMLHost para fazer referência ao objeto HTMLLoader atual.*

Tratamento de chamadas de JavaScript de window.moveBy(), window.moveTo(), window.resizeTo() e window.resizeBy()

Adobe AIR 1.0 e posterior

Substitua o método `set windowRect()` para tratar alterações nos limites do conteúdo HTML. O método `set windowRect()` é chamado quando o JavaScript em uma página chama `window.moveBy()`, `window.moveTo()`, `window.resizeTo()` ou `window.resizeBy()`. O exemplo a seguir simplesmente atualiza os limites da janela da área de trabalho:

```
override public function set windowRect(value:Rectangle):void
{
    htmlLoader.stage.nativeWindow.bounds = value;
}
```

Tratamento de chamadas JavaScript de window.open()

Adobe AIR 1.0 e posterior

Substitui o método `createWindow()` para tratar chamadas JavaScript de `window.open()`. As implementações do método `createWindow()` são responsáveis pela criação e retorno de um novo objeto HTMLLoader. Normalmente você exibe o HTMLLoader em uma nova janela, mas não é necessário criar uma nova janela.

O exemplo a seguir ilustra como implementar a função `createWindow()` usando `HTMLLoader.createRootWindow()` para criar a janela e o objeto HTMLLoader. Você também pode criar o objeto NativeWindow separadamente e adicionar o HTMLLoader ao palco da janela.

```
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;
}
```

Nota: *Esse exemplo atribui a implementação HTMLHost personalizada a qualquer janela nova criada com `window.open()`. Você também pode usar uma implementação diferente ou definir a propriedade `htmlHost` como null em novas janelas, se desejar.*

O objeto passado como parâmetro para o método `createWindow()` é um objeto HTMLWindowCreateOptions. A classe HTMLWindowCreateOptions inclui propriedades que informam os valores definidos na sequência de parâmetro `features` na chamada de `window.open()`:

propriedade HTMLWindowCreateOptions	Configuração correspondente na string de recursos na chamada JavaScript para window.open()
`fullscreen`	`fullscreen`
`height`	`height`
`locationBarVisible`	`location`
`menuBarVisible`	`menubar`
`resizeable`	`resizable`
`scrollBarsVisible`	`scrollbars`
`statusBarVisible`	`status`
`toolBarVisible`	`toolbar`
`width`	`width`
`x`	`left` ou `screenX`
`y`	`top` ou `screenY`

A classe HTMLLoader não implementa todos os recursos que podem ser especificados na string de recursos. O aplicativo deve fornecer barras de rolagem, barras de localização, barras de menu, barras de status e barras de ferramentas, quando apropriado.

Os outros argumentos para o método `window.open()` de JavaScript são tratados pelo sistema. A implementação `createWindow()` não deve carregar conteúdo no objeto HTMLLoader nem definir o título da janela.

Tratamento de chamadas JavaScript de window.close()

Adobe AIR 1.0 e posterior

Substitui o método `windowClose()` para tratar chamadas JavaScript do método `window.close()`. O exemplo a seguir fecha a janela da área de trabalho quando o método `window.close()` é chamado:

```
override public function windowClose():void
{
    htmlLoader.stage.nativeWindow.close();
}
```

As chamadas JavaScript de `window.close()` não têm que fechar as janelas que as contêm. Você pode, por exemplo, remover o HTMLLoader da lista de exibição, deixando a janela (que pode ter outro conteúdo) aberta, conforme o código a seguir:

```
override public function windowClose():void
{
    htmlLoader.parent.removeChild(htmlLoader);
}
```

Tratamento de alterações da propriedade windows.status

Adobe AIR 1.0 e posterior

Substitua o método `updateStatus()` para tratar alterações JavaScript no valor de `window.status`. O exemplo a seguir rastreia o valor do status:

```
 override public function updateStatus(status:String):void
{
    trace(status);
}
```

O status solicitado é passado como string para o método `updateStatus()`.

O objeto HTMLLoader não oferece uma barra de status.

Tratamento de alterações da propriedade window.document.title

Adobe AIR 1.0 e posterior

Substitua o método `updateTitle()` para tratar alterações JavaScript no valor de `window.document.title`. O exemplo a seguir altera o título da janela e acrescenta a sequência, "Sample", ao título:

```
 override public function updateTitle(title:String):void
{
    htmlLoader.stage.nativeWindow.title = title + " - Sample";
}
```

Quando `document.title` estiver definido em uma página HTML, o título solicitado será passado como string para o método `updateTitle()`.

As alterações em `document.title` não devem alterar o título da janela que contém o objeto HTMLLoader. Você pode, por exemplo, alterar outro elemento de interface, como campo de texto.

Tratamento de chamadas JavaScript de window.blur() e window.focus()

Adobe AIR 1.0 e posterior

Substitua os métodos `windowBlur()` e `windowFocus()` para tratar chamadas JavaScript de`window.blur()` e `window.focus()`, conforme mostrado o exemplo a seguir:

```
 override public function windowBlur():void
{
    htmlLoader.alpha = 0.5;
}
override public function windowFocus():void
{
    htmlLoader.alpha = 1.0;
    NativeApplication.nativeApplication.activate(htmlLoader.stage.nativeWindow);
}
```

Nota: *O AIR não fornece nenhuma API para desativar janela nem aplicativo.*

Criação de janelas com conteúdo HTML de rolagem

Adobe AIR 1.0 e posterior

A classe HTMLLoader inclui um método estático, `HTMLLoader.createRootWindow()`, que permite abrir uma nova janela (representada por um objeto NativeWindow) que contém um objeto HTMLLoader e define algumas configurações da interface do usuário para essa janela. O método tem quatro parâmetros que permitem definir a interface do usuário:

Parâmetro	Descrição
`visible`	Um valor booliano que especifica se a janela ficará inicialmente visível (`true`) ou não (`false`).
`windowInitOptions`	Um objeto NativeWindowInitOptions. A classe NativeWindowInitOptions define as opções de inicialização do objeto NativeWindow, incluindo o seguinte: se a janela é minimizável, maximizável ou redimensionável, se a janela tem cromo do sistema ou cromo personalizado, se a janela é transparente ou não (para janelas que não usam o cromo do sistema) e o tipo de janela.
`scrollBarsVisible`	Se há barras de rolagem (`true`) ou não (`false`).
`bounds`	Um objeto Rectangle definindo a posição e as dimensões da nova janela.

Por exemplo, o código a seguir usa o método `HTMLLoader.createRootWindow()` para criar uma janela com conteúdo HTMLLoader que usa barras de rolagem:

```
 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();
```

Nota: *Janelas criadas chamando* `createRootWindow()` *diretamente em JavaScript permanecem independentes da janela HTML aberta. As propriedades* `opener` *e* `parent` *da janela JavaScript, por exemplo, são* `null`*. No entanto, se você chamar* `createRootWindow()` *indiretamente substituindo o método* `createWindow()` *do HTMLHost para chamar* `createRootWindow()`*,* `opener` *e* `parent` *farão referência à janela HTML aberta.*

Criação de subclasses da classe HTMLLoader

Adobe AIR 1.0 e posterior

Você pode criar uma subclasse da classe HTMLLoader para criar novos comportamentos. Por exemplo, você pode criar uma subclasse que defina ouvintes de evento padrão de eventos HTMLLoader (como os eventos despachados quando o HTML é processado ou quando um usuário clica em um link).

O exemplo a seguir estende a classe HTMLHost para apresentar comportamento *normal* quando o método `window.open()` de JavaScript for chamado. O exemplo a seguir define uma subclasse do HTMLLoader que usa a classe personalizada de implementação HTMLHost:

```
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
       }
}
```

O seguinte define uma subclasse da classe HTMLLoader que atribui um objeto MyHTMLHost à respectiva propriedade `htmlHost`:

```
package
{
       import flash.html.HTMLLoader;
    import MyHTMLHost;
    import HTMLLoader;
       public class MyHTML extends HTMLLoader
       {
           public function MyHTML()
           {
               super();
               htmlHost = new MyHTMLHost();
           }
       }
}
```

Para obter detalhes sobre a classe HTMLHost e o método `HTMLLoader.createRootWindow()` usado nesse exemplo, consulte "Definição de interfaces do usuário como navegadores para conteúdo HTML" na página 1044.

Capítulo 61: Tratamento de eventos relacionados a HTML no AIR

Adobe AIR 1.0 e posterior

O sistema de tratamento de eventos permite que os programadores respondam às entradas de usuários e eventos do sistema de forma conveniente. O modelo de evento do Adobe® AIR® não é apenas conveniente, mas também compatível com os padrões. Com base na Especificação de eventos DOM (Modelo de objeto de documento) de Nível 3, uma arquitetura de tratamento de eventos padrão do segmento, o modelo de evento oferece uma ferramenta de tratamento de eventos eficiente e intuitiva para programadores.

eventos HTMLLoader

Adobe AIR 1.0 e posterior

O objeto HTMLLoader despacha os seguintes eventos do Adobe® ActionScript®3.0:

Evento	Descrição
`htmlDOMInitialize`	Despachado quando o documento HTML é criado, mas antes que qualquer script seja analisado ou que os nós DOM sejam adicionados à página.
`complete`	Despachado quando o HTML DOM tiver sido criado em resposta à operação de carregamento, logo após o evento `onload` na página HTML.
`htmlBoundsChanged`	Despachado quando uma ou as duas propriedades `contentWidth` e `contentHeight` são alteradas.
`locationChange`	Despachado quando a propriedade location do HTMLLoader foi alterada.
`locationChanging`	Despachado antes que o local do HTMLLoader seja alterado por causa da navegação do usuário, de uma chamada do JavaScript ou de um redirecionamento. O evento `locationChanging` não é despachado quando os métodos `load()`, `loadString()`,`reload()`,`historyGo()`,`historyForward()` e `historyBack()` são chamados. A chamada do método `preventDefault()` do objeto event despachado cancela a navegação. Se um link for aberto no navegador do sistema, um evento locationChanging não será despachado até que HTMLLoader não seja mudado de local.
`scroll`	Despachado a sempre que o mecanismo HTML altera a posição de rolagem. Eventos de rolagem podem ocorrer devido à navegação para ancorar links (`n°` de links) na página ou devido às chamadas do método`window.scrollTo()`. Inserir texto em uma entrada de texto ou área de texto também pode gerar um evento de rolagem.
`uncaughtScriptException`	Despachado quando ocorre uma exceção JavaScript no HTMLLoader e a exceção não é capturada no código JavaScript.

Você também pode registrar uma função ActionScript para o evento JavaScript (como `onClick`). Para obter detalhes, consulte "Tratamento de eventos DOM com o ActionScript" na página 1058.

Tratamento de eventos DOM com o ActionScript

Adobe AIR 1.0 e posterior

Você pode registrar as funções do ActionScript para que respondam a eventos JavaScript. Por exemplo, considere o seguinte conteúdo HTML:

```
<html>
<body>
    <a href="#" id="testLink">Click me.</a>
</html>
```

Você pode registrar uma função do ActionScript como manipulador de qualquer evento na página. Por exemplo, o código a seguir adiciona a função `clickHandler()` como o ouvinte do evento `onclick` do elemento `testLink` na página HTML:

```
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 );
}
```

O objeto de evento despachado não é do tipo flash.events.Event nem uma das subclasses Event. Use a classe Object para declarar um tipo para o argumento da função de tratamento de evento.

Você também pode usar o método `addEventListener()` para se registrar para esses eventos. Por exemplo, você pode substituir o método `completeHandler()` no exemplo anterior pelo seguinte código:

```
function completeHandler(event:Event):void {
    var testLink:Object = html.window.document.getElementById("testLink");
    testLink.addEventListener("click", clickHandler);
}
```

Quando um ouvinte se refere a um elemento DOM específico, é bom aguardar que o HTMLLoader pai despache o evento `complete` antes de adicionar os ouvintes de evento. As páginas HTML com frequência carregam vários arquivos e o HTML DOM não é criado totalmente até que todos os arquivos sejam carregados e analisados. O HTMLLoader despacha o evento `complete` quando todos os elementos tiverem sido criados.

Resposta a exceções JavaScript não capturadas

Adobe AIR 1.0 e posterior

Considere o seguinte HTML:

```
<html>
<head>
    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

Ele contém uma função JavaScript, `throwError()`, que faz referência a uma variável desconhecida, `melbaToast`:

```
var x = 400 * melbaToast;
```

Quando a operação de JavaScript encontra uma operação ilegal não capturada no código JavaScript com uma estrutura `try/catch`, o objeto HTMLLoader que contém a página despacha um evento HTMLUncaughtScriptExceptionEvent. Você pode registrar um manipulador para esse evento, como no código a seguir:

```
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);
    }
}
```

No JavaScript, você pode tratar o mesmo evento usando a propriedade window.htmlLoader:

```
<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>
```

O manipulador de eventos `htmlErrorHandler()` cancela o comportamento padrão do evento (que é enviar a mensagem de erro de JavaScript para a saída trace do AIR) e gera sua própria mensagem de saída. Ele produz o valor do `exceptionValue` do objeto HTMLUncaughtScriptExceptionEvent. Ele produz as propriedades de cada objeto na matriz `stackTrace`:

```
exceptionValue: ReferenceError: Can't find variable: melbaToast
sourceURL: app:/test.html
line: 5
function: throwError
sourceURL: app:/test.html
line: 10
function: onclick
```

Tratamento de eventos de tempo de execução com JavaScript

Adobe AIR 1.0 e posterior

As classes de tempo de execução oferecem suporte à adição de manipuladores de eventos com o método `addEventListener()`. Para adicionar uma função do manipulador a um evento, chame o método `addEventListener()` do objeto que despacha o evento, fornecendo o tipo de evento e a função de tratamento. Por exemplo, para ouvir o evento `closing` despachado quando o usuário clica no botão Fechar da janela na barra de título, use a seguinte instrução:

```
window.nativeWindow.addEventListener(air.NativeWindow.CLOSING, handleWindowClosing);
```

Criação de função do manipulador de eventos

Adobe AIR 1.0 e posterior

O código a seguir cria um arquivo HTML simples que exibe informações sobre a posição da janela principal. A função do manipulador, chamada de `moveHandler()`, ouve o evento move (definido pela classe NativeWindowBoundsEvent) da janela principal.

```
<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>
```

Quando o usuário move a janela, os elementos da área de texto exibem as posições X e Y atualizadas da janela:

Observe que o objeto de evento é passado como argumento para o método `moveHandler()`. O parâmetro event permite que a função do manipulador examine o objeto de evento. Neste exemplo, você usa a propriedade `type` do objeto de evento para informar que o evento é um evento `move`.

Remoção de ouvintes de eventos

Adobe AIR 1.0 e posterior

Você pode usar o método `removeEventListener()` para remover um ouvinte de evento que não seja mais necessário. É uma boa ideia remover o ouvinte que não será mais usado. Entre os parâmetros obrigatórios estão `eventName` e `listener`, que são os mesmos parâmetros obrigatórios no método `addEventListener()`.

Remoção de ouvintes de eventos nas páginas HTML que navegam

Adobe AIR 1.0 e posterior

Quando o conteúdo HTML é navegado ou quando é descartado porque a janela que o contém é fechada, os ouvintes de eventos que fazem referência a objetos na página não carregada não são removidos automaticamente. Quando um objeto envia um evento para um manipulador que já foi descarregado, a mensagem de erro a seguir é exibida: "O aplicativo tentou criar uma referência a um objeto JavaScript e uma página HTML que não está mais carregada."

Para evitar esse tipo de erro, remova os ouvintes de eventos JavaScript da página HTML antes que ele parta. No caso de navegação de página (em um objeto HTMLLoader), remova o ouvinte de evento durante o evento `unload` do objeto `window`.

Por exemplo, o código JavaScript a seguir remove um ouvinte de evento de um evento `uncaughtScriptException`:

```
window.onunload = cleanup;
window.htmlLoader.addEventListener('uncaughtScriptException', uncaughtScriptException);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Para impedir que o erro ocorra ao fechar janelas que contêm esse conteúdo HTML, chame uma função de limpeza em resposta ao evento `closing` do objeto NativeWindow (`window.nativeWindow`). Por exemplo, o código JavaScript a seguir remove um ouvinte de evento de um evento `uncaughtScriptException`:

```
 window.nativeWindow.addEventListener(air.Event.CLOSING, cleanup);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Você também pode impedir esse erro removendo um ouvinte de evento tão logo seja executado (se o evento só precisar ser tratado uma vez). Por exemplo, o código JavaScript a seguir cria uma janela html chamando o método `createRootWindow()` da classe HTMLLoader e adiciona um ouvinte de evento ao evento `complete`. Quando o manipulador de eventos `complete` é chamado, ele remove o próprio ouvinte de evento usando a função `removeEventListener()`:

```
 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"));
```

Remover ouvintes de evento desnecessários também permite que o coletor de lixo do sistema recupere qualquer memória associada a esses ouvintes.

Verificação da existência de ouvintes de eventos

Adobe AIR 1.0 e posterior

O método `hasEventListener()` permite verificar a existência de um ouvinte de evento em um objeto.

Capítulo 62: Exibição de conteúdo HTML em aplicativos móveis

Adobe AIR 2.5 e posterior

A classe StageWebView exibe conteúdo HTML usando o controle do navegador do sistema em dispositivos móveis e usando o controle Adobe® AIR® HTMLLoader padrão em computadores desktop. Verifique a propriedade `StageWebView.isSupported` para determinar se a classe conta com suporte no dispositivo atual. O suporte não é garantido para todos os dispositivos no perfil móvel.

Em todos os perfis, a classe StageWebView oferece suporte somente à interação limitada entre o conteúdo HTML e o resto do aplicativo. Você pode controlar a navegação, mas scripts cruzados e a troca direta de dados não são permitidos. Você pode carregar conteúdo de um URL local ou remoto ou passá-lo em uma string de HTML.

Objetos StageWebView

O objeto StageWebView não é um objeto de exibição e não pode ser adicionado à lista de exibição. Em vez disso, ele funciona como um viewport anexado diretamente ao palco. O conteúdo StageWebView desenha sobre qualquer conteúdo de lista de exibição. Não há nenhum modo de controlar a ordem de desenho de diferentes objetos StageWebView.

Para exibir um objeto StageWebView, atribua o palco em que o objeto se encontra de modo a ser exibido para a propriedade `stage` do StageWebView. Defina o tamanho da exibição usando a propriedade `viewPort`.

Defina as coordenadas X e Y da propriedade `viewPort` entre -8192 e 8191. O valor máximo da altura e da largura do palco é 8191. Se o tamanho exceder os valores máximos, uma exceção será lançada.

O seguinte exemplo cria um objeto StageWebView, define as propriedades `stage` e `viewPort` e exibe uma string de HTML:

```
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 );
```

Para ocultar um objeto StageWebView, defina sua propriedade `stage` como `null`. Para destruir o objeto completamente, chame o método `dispose()`. Chamar `dispose()` é opcional, mas ajuda o coletor de lixo a recuperar mais cedo a memória usada pelo objeto.

Conteúdo

Você pode carregar conteúdo em um objeto StageWebView usando dois métodos: `loadURL()` e `loadString()`.

O método `loadURL()` carrega um recurso no URL especificado. Você pode usar qualquer esquema de URI com suporte no controle de navegador da Web do sistema, inclusive: data:, file:, http:, https: e javascript:. Os esquemas app: e app-storage: não são compatíveis. O AIR não valida a string do URL.

O método `loadString()` carrega uma string literal com conteúdo HTML. O local de uma página carregada com esse método é expresso como:

- No desktop: sobre:em branco
- No iOS: *htmlString*
- No Android: o formato do URI dos dados do *htmlString* codificado

O esquema de URI determina as regras de carregamento de conteúdo ou dados incorporados.

Esquema de URI	Recurso local local	Recurso remoto local	XMLHttpRequest local	XMLHttpRequest remoto
dados:	Não	Sim	Não	Não
arquivo:	Sim	Sim	Sim	Sim
http:, https:	Não	Sim	Não	Mesmo domínio
sobre: (método loadString())	Não	Sim	Não	Não

Nota: *Se a propriedade `displayState` do palco for definida como `FULL_SCREEN`, no desktop, você não poderá digitar em um campo de texto exibido em StageWebView. No entanto, no iOS e no Android, você pode digitar em uma campo de texto em StageWebView, mesmo que o `displayState` do palco seja `FULL_SCREEN`.*

O exemplo a seguir usa um objeto StageWebView para exibir o site da Adobe:

```
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" );
        }
    }
}
```

Em dispositivos Android, é preciso especificar a permissão INTERNET do Android para que o aplicativo carregue com sucesso os recursos remotos.

No Android 3.0+, um aplicativo deve habilita a aceleração de hardware no elemento Android manifestAdditions do descritor do aplicativo do AIR para exibir o conteúdo do plug-in num objeto StageWebView. Consulte Como Habilitar o Flash Player e outros plug-ins em um objeto StageWebView.

URI JavaScript

Você pode usar um URI JavaScript para chamar uma função definida na página HTML carregada pelo objeto StageWebView. A função que você chama usando o URI JavaScript URI é executada no contexto da página da Web carregada. O exemplo a seguir usa um objeto StageWebView para chamar uma função JavaScript:

```
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 );
        }
    }
}
```

Mais tópicos da Ajuda

Sean Voisen: Making the Most of StageWebView (Aproveitando ao Máximo o StageWebView)

Eventos de navegação

Quando o usuário clica em um link no HTML, o objeto StageWebView despacha um evento `locationChanging`. Você pode chamar o método `preventDefault()` do objeto de evento para parar a navegação. Caso contrário, o objeto StageWebView navegará para a nova página e despachará um evento `locationChange`. Quando o carregamento da página é concluído, o StageWebView despacha um evento `complete`.

Um evento `locationChanging` é despachado a cada redirecionamento do HTML. Os eventos `locationChange` e `complete` são despachados no momento oportuno.

No iOS, um evento `locationChanging` é despachado antes de um evento `locationChange`, com exceção dos primeiros métodos `loadURL()` ou `loadString()`. Um evento `locationChange` também é despachado para alterações de navegação por iFrames e Frames.

O exemplo a seguir ilustra como você pode evitar uma alteração local e, em vez disso, abrir abrir a nova página no navegador do sistema.

```
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 ) );
        }
    }
}
```

Histórico

Quando o usuário clica em links no conteúdo exibido em um objeto StageWebView, o controle salva as pilhas de histórico de avanço e de retrocesso. O exemplo a seguir ilustra como navegar pelas duas pilhas de histórico. O exemplo usa as teclas programáveis Voltar e Pesquisar.

```
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();
            }
        }
    }
}
```

Foco

Embora não seja um objeto de exibição, a classe StageWebView contém membros que permitem gerenciar as transições do foco para um controle ou para fora dele.

Quando o objeto StageWebView recebe o foco, despacha um evento `focusIn`. Use esse evento para gerenciar os elementos de foco no aplicativo, se necessário.

Quando o StageWebView abre mão do foco, despacha um evento `focusOut`. Uma ocorrência de StageWebView pode abrir mão do foco quando o usuário usa a tabulação passando pelo primeiro ou pelo último controle na página da Web usando o trackball ou as setas de direção de um dispositivo. A propriedade `direction` do objeto de evento permite saber se o fluxo do foco está subindo acima do topo da página ou descendo além da parte inferior da página. Use estas informações para atribuir o foco ao objeto de exibição apropriado acima ou abaixo de StageWebView.

No iOS, o foco não pode ser definido programaticamente. O StageWebView despacha eventos `focusIn` e `focusOut` com a propriedade direction de `FocusEvent` definida como `none`. Se o usuário tocar dentro do StageWebView, o evento `focusIn` é despachado. Se o usuário tocar fora do StageWebView, o evento `focusOut` é despachado.

O exemplo a seguir ilustra como o foco passa do objeto StageWebView para objetos de exibição do Flash:

```
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;
        }

    }
}
```

Captura de bitmap

Um objeto StageWebView é renderizado acima de todo o conteúdo da lista de exibição. Não é possível adicionar um conteúdo acima do objeto StageWebView. Por exemplo, não é possível expandir uma lista suspensa sobre o conteúdo de StageWebView. Para solucionar esse problema, faça a captura de um instantâneo de StageWebView. Em seguida, oculte o StageWebView e adicione o instantâneo em bitmap em seu lugar.

O exemplo a seguir ilustra como capturar o instantâneo de um objeto StageWebView usandop o método `drawViewPortToBitmapData`. Ele oculta o objeto StageWebView definido o palco como null. Depois que a página da Web estiver totalmente carregada, chame uma função que captura o bitmap e o exibe. Quando executado, o código exibe dois rótulos, Google e Facebook. Clicar no rótulo captura a página da Web correspondente e a exibe como um instantâneo no palco.

```
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);
        }
    }
}
```

Capítulo 63: Usar workers para simultaneidade

Flash Player 11.4 e posterior, Adobe AIR 13.4 e posterior para plataformas desktop

Os workers do ActionScript geralmente tornam possível executar o código simultaneamente, em outras palavras, executar o código em segundo plano sem interromper a execução do código principal.

As APIs de simultaneidade do ActionScript estão disponíveis somente em Flash Player 11.4 ou posterior e AIR 3.4 e posterior, nas plataformas desktop. A simultaneidade não é suportada no AIR para plataformas móveis.

Entendendo workers e simultaneidade

Flash Player 11.4 e posterior, Adobe AIR 13.4 e posterior para plataformas desktop

Quando um aplicativo não usa workers, o código do aplicativo é executado em um bloco linear único de etapas de execução conhecido como uma cadeia de *execução*. A cadeia executa o código escrito pelo desenvolvedor. Executa também a maioria dos códigos que fazem parte do runtime, notavelmente o código que atualiza a tela quando mudam as propriedades dos objetos de exibição. Apesar dos códigos serem escritos em blocos como métodos e classes, durante o runtime, o código executa uma linha por um tempo como se por vez ainda estivessem escritos em uma única longa série de etapas. Considerando este exemplo hipotético de etapas que um aplicativo executa:

1. Inserir quadro: o runtime chama quaisquer manipuladores de eventos `enterFrame` e executa um código de cada vez
2. Evento de mouse: o usuário movimenta o mouse e o runtime chama quaisquer manipuladores de eventos de mouse conforme os vários eventos rollover e rollout acontecem
3. Carregar evento completo: uma solicitação para carregar um arquivo xml de uma url retorna com os dados de arquivo carregados. O manipulador de eventos é chamado e executado por etapas, lendo o conteúdo xml e criando um conjunto de objetos com base nos dados xml.
4. Eventos de mouse: o mouse é movido novamente, assim o runtime chama a manipulação de eventos de mouse relevante
5. Renderização: não há mais eventos aguardando, por isso o runtime atualiza a tela com base nas alterações feitas aos objetos de exibição
6. Inserir quadro: reinicia o ciclo

Conforme descrito no exemplo, as etapas hipotéticas de 1 a 5 é executada na sequência dentro de um único bloco de tempo chamado de quadro. Como eles são executados em sequência em uma cadeia única, o runtime não pode interromper uma etapa do processo para executar outra. Em uma taxa de quadros de 30 quadros por segundo, a execução possui menos de um trigésimo de segundo para executar todas as operações. Em alguns casos, é tempo suficiente para executar o código e o runtime apenas aguarda durante o tempo restante. Entretanto, suponha que os

dados xml carregados na etapa 3 sejam de uma estrutura xml muito grande e profundamente aninhada. Como o código é repetido no xml e cria objetos, provavelmente pode levar mais que um trigésimo de segundo para realizar este trabalho. Neste caso, as próximas etapas (responder ao mouse e redesenhar a tela) não acontecem logo como deveriam. Isso causa o congelamento e o corte na tela como se ela não fosse redesenhada rápido o suficiente em resposta ao movimento do mouse pelo usuário.

Se todos os códigos forem executados na mesma cadeia, só há um modo de evitar cortes e congelamentos ocasionais. Isso serve para não realizar operações longas como um loop em um grande conjunto de dados. Os workers do ActionScript fornecem outra solução. Usando um worker, você pode executar códigos longos em um worker separado. Cada worker é executado em uma cadeia separada para que o worker de segundo plano movimente a operação longa em sua própria cadeia. Isso libera a cadeia de execução do worker principal para redesenhar a tela a cada quadro sem ser bloqueada por outro trabalho.

A capacidade de executar várias operações de código ao mesmo tempo dessa forma é conhecida como *simultaneidade*. Quando o worker de segundo plano finaliza o seu trabalho, ou está em pontos de "andamento" durante o processo, você pode enviar notificações e dados do worker principal. Deste modo, você pode escrever o código que realiza operações complexas ou demoradas, evitando uma má experiência de usuário de ter a tela congelada.

Workers são úteis, pois diminuem as chances de uma queda na taxa de quadros devido ao bloqueio da cadeia de renderização principal por outro código. No entanto, os workers requerem um sistema de memória adicional e o uso de CPU, que podem afetar o desempenho geral do aplicativo. Como cada worker utiliza sua própria instância da máquina de runtime virtual, mesmo a sobrecarga de um worker trivial pode ser grande. Ao utilizar workers, teste o seu código em todas as plataformas de destino para certificar-se de que as demanda do sistema não são tão grandes. A Adobe recomenda que você não use mais de um ou dois workers de segundo plano em um cenário típico.

Criando e gerenciando workers

Flash Player 11.4 e posterior, Adobe AIR 13.4 e posterior para plataformas desktop

A primeira etapa no uso do worker para simultaneidade é criar um worker de segundo plano. Use dois tipos de objetos para criar um worker. O primeiro é uma instância do worker, que é o que você criou. A outra é um objeto WorkerDomain, que cria o worker e gerencia a execução de objetos do worker em um aplicativo.

Quando o runtime é carregado, ele cria automaticamente o objeto WorkerDomain. A execução também cria automaticamente um worker para o swf principal do aplicativo. O primeiro worker é conhecido como o*worker primordial*.

Como há somente um objeto WorkerDomain para um aplicativo, você acessa a instância do WorkerDomain utilizando a propriedade estática `WorkerDomain.current` .

A qualquer momento, você pode acessar a instância atual do worker (o worker no qual o código atual está sendo executado) utilizando a propriedade estática `Worker.current` .

Criar um objeto worker de um swf

Assim como o swf principal é executado dentro do worker primordial, o worker de segundo plano executa o código de um único arquivo swf. Para usar um worker de segundo plano, você deve criar e compilar o código do worker como um arquivo swf. Para criar um worker de segundo plano, o worker pai precisa acessar os bytes dos arquivos swf como um objeto ByteArray. Passe o ByteArray para o método `createWorker()` do objeto WorkerDomain para criar o worker.

Há três modos principais de ter o worker de segundo plano swf como um objeto ByteArray:

Incorporar o swf do worker

Usar a metatag [Embed] para incorporar o swf do worker em um swf principal como um ByteArray:

```
[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
}
```

O swf do worker é compilado em um swf principal como uma subclasse do ByteArray nomeado como BgWorker_ByteClass. Criar uma instância dessa classe oferece a você um ByteArray previamente preenchido com os bytes do swf do worker.

Carregando um swf externo do worker

Usar um objeto URLLoader para carregar um arquivo swf externo. O arquivo swf precisa vir com o mesmo domínio de segurança, assim como um arquivo swf carregado do mesmo domínio da internet como o swf principal ou incluso no pacote de aplicativo AIR.

```
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
}
```

Quando o URLLoader finaliza o carregamento do arquivo swf, os bytes do swf ficam disponíveis na propriedade `data` do objeto URLLoader (`event.target.data` no exemplo).

Usando o swf principal como o swf do worker

Você pode usar um único swf como o swf principal e o swf do worker. Usar a propriedade `loaderInfo.bytes` da classe de exibição principal para acessar os bytes do swf.

```
// 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
    }
}
```

Se você utilizar essa técnica, use uma instrução `if` para ramificar o código do arquivo swf dentro do construtor da classe principal ou um método que ele chamar. Para determinar se o código está em execução no worker principal ou no worker de segundo plano, verifique a propriedade `isPrimordial` do objeto do worker atual, como apresentado no exemplo.

Iniciar uma execução do worker

Depois de criar um worker, inicie a execução do código chamando o método `start()` do objeto do worker. A operação `start()` não acontece imediatamente. Para saber quando o worker está em execução, registre um ouvinte para o evento `workerState` do objeto do worker. O evento é despachado quando o objeto do worker alterna estados em seu ciclo de vida, como quando inicia a execução de código. No manipulador de eventos `workerState`, verifique se a propriedade `state` do objeto do worker é `WorkerState.RUNNING`. Nesse ponto, o worker está sendo executado e o construtor da classe principal dele foi executado. A lista de códigos a seguir apresenta um exemplo de registro para o evento `workerState` e chamada do método `start()`:

```
// 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.
    }
}
```

Gerenciar execução do worker

A qualquer momento você pode acessar o conjunto de workers em execução em seu aplicativo usando o método `listWorkers()` da classe WorkerDomain. Esse método retorna o conjunto de workers cuja propriedade `state` é `WorkerState.RUNNING`, incluindo do worker primordial. Se um worker não foi iniciado ou se sua execução já foi interrompida, ele não será incluído.

Se você não precisar mais de um worker, poderá chamar o método `terminate()` do objeto do worker para fechá-lo e liberar a memória e outros recursos do sistema.

Comunicação entre workers

Flash Player 11.4 e posterior, Adobe AIR 13.4 e posterior para plataformas desktop

Apesar de workers executarem seus códigos em cadeias de execução separadas, eles não ofereceriam qualquer benefício se estivessem completamente isolados uns dos outros. Em resumo, a comunicação entre workers significa transmitir dados entre workers. Existem três mecanismos principais para levar dados de um worker para o outro.

Ao decidir qual dessas técnicas de compartilhamento de dados é apropriada para uma necessidade específica de transmissão de dados, considere as duas principais diferenças entre elas. Uma diferença entre elas é se há um evento para notificar o destinatário sobre a disponibilidade de novos dados ou se ele deve verificar se há atualizações. Outra diferença entre essas técnicas de compartilhamento de dados está relacionada à maneira com a qual os dados são transmitidos. Em alguns casos, o worker destinatário recebe uma cópia dos dados compartilhados, o que significa que mais objetos são criados, ocupando mais memórias e ciclos da CPU. Em outros casos, eles acessam os objetos que fazem referência à memória do mesmo sistema subjacente, o que significa que menos objetos são criados e, de maneira geral, menos memória é utilizada. Essas diferenças são descritas aqui:

Técnica de comunicação	Envia um evento ao receber dados	Compartilha memória entre workers
Propriedades compartilhadas entre workers	Não	Não, os objetos são cópias, não referências
MessageChannel	Sim	Não, os objetos são cópias, não referências
ByteArray compartilhável	Não	Sim, a memória é compartilhada

Transmitir dados com uma propriedade compartilhada

O modo mais básico de compartilhar dados entre workers é utilizando uma propriedade compartilhada. Cada worker mantém um dicionário interno de valores de propriedade compartilhados. As propriedades são armazenadas com uma nomes chave de sequência para fins de distinção entre elas. Para armazenar um objeto em um worker como uma propriedade compartilhada, chame o método `setSharedProperty()` do objeto do worker com dois argumentos, o nome chave e o valor a armazenar:

```
// code running in the parent worker
bgWorker.setSharedProperty("sharedPropertyName", someObject);
```

Uma vez que a propriedade compartilhada esteja definida, o valor poderá ser lido chamando o método `getSharedProperty()` do objeto do worker, transferindo o nome chave:

```
// code running in the background worker
receivedProperty = Worker.current.getSharedProperty("sharedPropertyName");
```

Não há restrição a respeito de qual worker lê ou define o valor da propriedade. Por exemplo, o código em um worker de segundo plano pode chamar o método `setSharedProperty()` para armazenar um valor. O código executado em um worker pai pode usar `getSharedProperty()` para receber os dados.

O valor transferido para o método `setSharedProperty()` pode ser praticamente qualquer tipo de objeto. Ao chamar o método `getSharedProperty()`, o objeto que retorna é uma cópia do objeto transferido para `setSharedProperty()` e não uma referência para o mesmo objeto, exceto em alguns casos especiais. Os princípios de como dados são compartilhados são explicados em "Referências compartilhadas e valores copiados" na página 1077.

A maior vantagem de utilizar uma propriedade compartilhada para transferir dados entre workers é a disponibilidade mesmo antes do worker ser executado. Você pode chamar um método `setSharedProperty()` do objeto do worker de segundo plano para definir uma propriedade compartilhada mesmo antes da execução do worker. Quando o worker pai chama o método `start()` do worker, o runtime chama o construtor da classe principal do worker filho. Quaisquer propriedades compartilhadas definidas antes de chamar `start()` estarão disponíveis para código no worker filho.

Transferindo dados com um MessageChannel

Um canal de mensagens fornece um link unidirecional de transferência de dados entre dois workers. Usar um objeto MessageChannel para transferir dados entre workers tem uma vantagem fundamental. Quando você envia uma mensagem (um objeto) utilizando um canal de mensagem, o objeto MessageChannel despacha um evento `channelMessage` . O código no worker de destino ouvir este evento para saber quando os dados estão disponíveis. Dessa forma, o worker de destino não precisa verificar continuamente se há atualizações de dados.

Um canal de mensagem é associado a apenas dois workers, um de origem e um de destino. Para criar um objeto MessageChannel, chame o método `createMessageChannel()` do objeto do worker de origem, transmitindo o worker de destino como um argumento:

```
// In the sending worker swf
var sendChannel:MessageChannel;
sendChannel = Worker.current.createMessageChannel(receivingWorker);
```

Ambos workers precisam ter acesso ao objeto MessageChannel. O modo mais simples de fazer isso é transferindo o objeto MessageChannel usando o método `setSharedProperty()` :

```
receivingWorker.setSharedProperty("incomingChannel", sendChannel);
```

No worker de destino, registre um ouvinte para o evento `channelMessage` do objeto MessageChannel. Esse evento é despachado quando o worker de origem envia dados pelo canal de mensagens.

```
// In the receiving worker swf
var incomingChannel:MessageChannel;
incomingChannel = Worker.current.getSharedProperty("incomingChannel");
incomingChannel.addEventListener(Event.CHANNEL_MESSAGE, handleIncomingMessage);
```

Para realmente enviar dados, chame o método `send()` do objeto MessageChannel no worker de origem:

```
// In the sending worker swf
sendChannel.send("This is a message");
```

No worker de destino, o MessageChannel chama o manipulador de eventos `channelMessage` . O worker de destino pode obter os dados chamando o método `receive()` do objeto MessageChannel.

```
private function handleIncomingMessage(event:Event):void
{
    var message:String = incomingChannel.receive() as String;
}
```

O objeto retornado pelo método de destino possui o mesmo tipo de dados do objeto que foi enviado para o método `send()` . O objeto de destino é uma cópia do objeto enviado pelo remetente e não uma referência para o objeto no worker de origem, a menos que seja um de alguns tipos de dados, como descrito em "Referências compartilhadas e valores copiados" na página 1077.

Compartilhamento de dados usando um ByteArray compartilhável

Quando um objeto é transmitido entre dois workers, o worker destinatário recebe um novo objeto que é uma cópia do original. Os dois objetos são armazenados em locais diferentes da memória do sistema. Consequentemente, cada cópia do objeto recebida aumenta o total de memória usado pelo runtime. Além disso, qualquer alteração realizada em um objeto de um worker não afeta a cópia do outro worker. Para obter mais detalhes sobre como os dados são copiados, consulte "Referências compartilhadas e valores copiados" na página 1077.

Por padrão, um objeto ByteArray usa o mesmo comportamento. Se você transmitir uma instância do ByteArray para o método `setSharedProperty()` do objeto de um worker ou um método `send()` de um objeto MessageChannel, o runtime cria um novo ByteArray na memória do computador e o worker destinatário recebe uma instância do ByteArray que é uma referência ao novo ByteArray. Entretanto, você pode alterar esse comportamento de um objeto ByteArray definindo sua propriedade `shareable` como `true`.

Quando um objeto ByteArray compartilhável é transmitido entre workers, a instância do ByteArray do worker destinatário é uma referência à memória do mesmo sistema operacional subjacente usada pela instância do ByteArray no worker remetente. Quando o código de um worker altera o conteúdo da matriz de bytes, essas alterações se tornam imediatamente disponíveis em outros workers com acesso à matriz de bytes compartilhada.

Como os workers executam seu código simultaneamente, dois workers podem tentar acessar os mesmos bytes em uma matriz de bytes ao mesmo tempo. Isso pode causar perda ou corrupção dos dados. Diversas APIs podem ser usadas para gerenciar o acesso aos recursos compartilhados e evitar esse tipo de problema.

A classe ByteArray tem métodos que permitem que você valide e altere o conteúdo da matriz de bytes em uma só operação:

- Método atomicCompareAndSwapIntAt()
- Método atomicCompareAndSwapLength()

Além disso, o pacote flash.concurrent inclui classes que fornecem controle de acesso para trabalhar com recursos compartilhados:

- Classe Mutex
- Classe Condition

Referências compartilhadas e valores copiados

No caso normal, quando você chama o `Worker.setSharedProperty()` ou o `MessageChannel.send()`,o objeto que é transmitido ao worker de destino é serializado no formato AMF. Isso tem algumas consequências:

- O objeto que é criado no worker de origem quando o método `getSharedProperty()` é chamado é desserializado pelos bytes AMF. É uma cópia do objeto original e não uma referência ao objeto. Quaisquer alterações feitas ao objeto nos workers não serão alteradas na cópia que está no outro worker.

- Objetos que não podem ser serializados no formato AMF, como objetos de exibição, não podem ser transmitidos a um worker usando `Worker.setSharedProperty()` ou `MessageChannel.send()` .
- Para desserializar corretamente uma classe personalizada, a definição da classe deve ser registrada utilizando a função `flash.net.registerClassAlias()` ou metadados `[RemoteClass]` . O mesmo alias deve ser usado pelas versões da classe de ambos os workers.

Há cinco casos especiais de objetos que são realmente compartilhados em vez de copiados entre workers:

- Objetos Worker
- Objetos MessageChannel
- matriz de bytes compartilhável (um objeto ByteArray cuja propriedade `shareable` é `true`)
- Objetos Mutex
- Objetos Condition

Quando você transmite uma instância de um desses objetos utilizando o método `Worker.setSharedProperty()` ou o método `MessageChannel.send()`, cada worker possui uma referência para o mesmo objeto subjacente. Modificações feitas a uma instância em uma worker são imediatamente disponibilizadas em outros workers. Além disso, se você transmite a mesma instância de um desses objetos para um worker mais de uma vez, o runtime não cria uma nova cópia de objetos no worker de destino. Em vez disso, a mesma referência é reutilizada.

Técnicas adicionais de compartilhamento de dados

Além do mecanismo específico de workers para transmissão de dados, os workers também podem trocar dados usando as APIs existentes que suportam compartilhamento de dados entre dois aplicativos swf, como o seguinte:

- objetos compartilhados locais
- escrever dados em um arquivo em um worker e ler do arquivo em outro worker
- armazenar e ler os dados de um banco de dados SQLite

Quando você compartilha um recurso entre dois ou mais workers, geralmente você precisa evitar ter vários workers acessando o recurso ao mesmo tempo. Por exemplo, vários workers acessando um arquivo do sistema de arquivos local pode causar perda ou corrupção de dados e pode não ser suportado pelo sistema operacional.

Para se proteger contra problemas de acesso simultâneo, use as classes Mutex e Condition no pacote flash.concurrent para fornecer controle de acesso para trabalhar com recursos compartilhados.

Diferentemente de outros mecanismos de compartilhamento de dados, o mecanismo do banco de dados SQLite é projetado para acesso simultâneo e possui seu próprio suporte de transação incorporado. Vários workers podem acessar um banco de dados SQLite sem o risco de corromper os dados. Como os workers usam instâncias diferentes do SQLConnection, cada worker acessa o banco de dados em uma transação separada. Operações de manipulação de dados simultâneas não afetam a integridade dos dados.

Consulte também

"Trabalho com bancos de dados SQL locais no AIR" na página 737
Pacote flash.concurrent

Capítulo 64: Segurança

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Segurança é uma preocupação principal da Adobe, dos usuários, dos proprietários de site e dos desenvolvedores de conteúdo. Por esta razão, o Adobe® Flash® Player e o Adobe® AIR™ incluem um conjunto de regras e controles de segurança para proteger o usuário, o proprietário do site e o desenvolvedor do conteúdo. Esta discussão abrange o modelo de segurança dos arquivos SWF publicados com o ActionScript 3.0 e executados no Flash Player 9.0.124.0 ou posterior, além de arquivos SWF, HTML e JavaScript executados no AIR 1.0 ou posterior, salvo indicado de outra forma.

Esta discussão fornece uma visão geral de segurança; ele não tenta explicar de maneira abrangente todos os detalhes de implementação, cenários de uso ou ramificações do uso de determinadas APIs. Para obter uma discussão mais detalhada sobre conceitos de segurança do Flash Player, consulte o tópico "Segurança" do Centro de desenvolvedores do Flash Player, em www.adobe.com/go/devnet_security_br.

Visão geral da segurança da Plataforma Flash

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Uma boa parte do modelo de segurança utilizado pelos aplicativos Flash Player e AIR tem por base o domínio de origem dos arquivos SWF, HTML, mídias e outros recursos carregados. Código executável em um arquivo de um domínio específico da Internet, como www.example.com, sempre pode acessar todos os dados daquele domínio. Esses ativos são colocados no mesmo agrupamento de segurança, conhecido como uma *caixa de proteção de segurança*. (Para obter mais informações, consulte "Caixas de proteção de segurança" na página 1081.)

Por exemplo, um código do ActionScript em um arquivo SWF pode carregar arquivos SWF, bitmaps, áudio, arquivos de texto e qualquer outro ativo do seu próprio domínio. Além disso, o cruzamento de scripts entre dois arquivos do mesmo domínio sempre é permitido, desde que os dois arquivos sejam escritos usando o ActionScript 3.0. O *Cross-scripting* é a capacidade do código de um arquivo de usar o ActionScript para acessar propriedades, métodos e objetos definidos pelo código em outro arquivo SWF.

O cross-scripting não é suportado entre arquivos SWF escritos usando o ActionScript 3.0 e versões anteriores do ActionScript. No entanto esses arquivos podem se comunicar usando a classe LocalConnection. Além disso, a capacidade de um arquivo SWF de executar cross-script de arquivos SWF do ActionScript 3.0 de outros domínios e de carregar dados de outros domínios é proibida por padrão. No entanto esse acesso pode ser concedido com uma chamada para o método `Security.allowDomain()` no arquivo SWF carregado. Para obter mais informações, consulte "Cross-scripting" na página 1101.

Por padrão, as seguintes regras básicas de segurança sempre se aplicam:

- Recursos na mesma caixa de proteção de segurança sempre podem acessar um ao outro.
- Código executável em arquivos em uma caixa de proteção remota nunca acessam arquivos e dados locais.

Os aplicativos Flash Player e AIR consideram o seguinte como domínios individuais, e criam caixas de proteção individuais para cada um:

- `http://example.com`
- `http://www.example.com`

- `http://store.example.com`
- `https://www.example.com`
- `http://192.0.34.166`

Mesmo que um domínio nomeado, como http://example.com, mapeie para um endereço IP específico, como http://192.0.34.166, o aplicativo criará caixas de proteção de segurança separadas para cada um.

Existem dois métodos básicos que podem ser usados por um desenvolvedor para conceder acesso a um arquivo SWF a ativos das caixas de proteção diferente daquela do arquivo SWF:

- O método `Security.allowDomain()` (consulte "Controles de autor (desenvolvedor)" na página 1093)
- O arquivo de política da URL (consulte "Controles de site (arquivos de política)" na página 1089)

Nos modelos de segurança do Flash Player e do aplicativo do AIR, há uma distinção entre carregamento de conteúdo e extração ou acesso a dados. *Content* é definido como mídia, incluindo mídia visual que o aplicativo pode exibir, áudio, vídeo ou um arquivo SWF ou HTML que inclui mídia exibida. *Data* é definido como algo que é acessível apenas ao código Content e data são carregados de maneiras diferentes.

- Carregamento de conteúdo — Você pode carregar conteúdos utilizando classes, tais como Loader, Sound e NetStream; através de marcas MXML ao usar o Flex; ou através de marcas HTML em um aplicativo do AIR.
- Extração de dados—você pode extrair dados do conteúdo de mídia carregado usando objetos Bitmap, os métodos `BitmapData.draw()` e `BitmapData.drawWithQuality()`, a propriedade `Sound.id3` ou o método `SoundMixer.computeSpectrum()`. O método `drawWithQuality` está disponível no Flash Player 11.3 e superior; AIR 3.3 e superior.
- Acesso a dados — é possível acessar dados diretamente carregando-os de um arquivo externo (como um arquivo XML) usando classes, como as classes URLStream, URLLoader, FileReference, Socket e XMLSocket. O AIR oferece mais classes para carregamento de dados, tais como FileStream e XMLHttpRequest.

O modelo de segurança do Flash Player define diferentes regras para carregamento de conteúdo e acesso a dados. Em geral, há menos restrições no carregamento de conteúdo do que no acesso a dados.

Em geral, o conteúdo (arquivos SWF, bitmaps, arquivos mp3 e vídeos) pode ser carregado a partir de qualquer local, mas se o conteúdo for de um domínio diferente daquele do arquivo código ou conteúdo carregado, ele será particionado em uma caixa de proteção de segurança separada.

Há algumas barreiras no que diz respeito ao carregamento de conteúdo:

- Por padrão, arquivos SWF locais (carregados de um endereço que não seja da rede, como do disco rígido de um usuário) são classificados na caixa de proteção local com sistema de arquivos. Esses arquivos não podem carregar conteúdo da rede. Para obter mais informações, consulte "Caixas de proteção locais" na página 1081.
- Servidores de protocolo RTMP podem limitar o acesso ao conteúdo. Para obter mais informações, consulte"Conteúdo entregue usando servidores RMTP" na página 1101.

Se a mídia carregada for uma imagem, áudio ou vídeo, seus dados, como dados de pixel e dados de som, poderão ser acessados por um arquivo SWF fora de sua caixa de proteção de segurança apenas se o domínio daquele arquivo SWF tiver sido incluído em um arquivo de política de URL no domínio de origem da mídia. Para obter detalhes, consulte "Acesso à mídia carregada como dados" na página 1105.

Outras formas de dados carregados incluem arquivos de texto ou XML que são carregados com um objeto URLLoader. Novamente nesse caso, para acessar quaisquer dados de outra caixa de proteção de segurança, permissão deve ser concedida por meio de um arquivo de política de URL no domínio de origem. Para obter detalhes, consulte "Uso de URLLoader e URLStream" na página 1107.

Nota: *Os arquivos de Política nunca são obrigatórios para os códigos executados na caixa de proteção do aplicativo do AIR para carregar conteúdos ou dados remotos.*

Caixas de proteção de segurança

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os computadores clientes podem obter arquivos individuais que contêm código, conteúdo e dados de várias fontes, como sites externos, um sistema local de arquivos ou de um aplicativo do AIR instalado. Os aplicativos Flash Player e AIR atribuem arquivos de código e outros recursos individualmente, como objetos compartilhados, bitmaps, sons, vídeos e arquivos de dados, a caixas de proteção de segurança com base em sua origem quando eles são carregadas. As seções a seguir descrevem as regras aplicadas pelos aplicativos que controlam o que um código ou conteúdo dentro de uma determinada caixa de proteção pode acessar.

Para obter mais informações sobre a segurança do Flash Player, consulte o tópico "Segurança" do Centro de desenvolvedores do Flash Player, em www.adobe.com/go/devnet_security_br.

Caixas de proteção remotas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os aplicativos Flash Player e AIR classificam os recursos (inclusive arquivos SWF) da Internet em caixas de proteção separadas que correspondem ao seu domínio de origem. Por exemplo, os recursos carregados de *example.com* serão colocados em uma caixa de proteção diferente dos recursos carregados de *foo.org*. Por padrão, esses arquivos têm permissão para acessar quaisquer recursos de seu próprio servidor. Arquivos SWF remotos podem receber permissão para acessar dados adicionais de outros domínios por permissões explícitas de autor e de site, como arquivos de política de URL e o método `Security.allowDomain()`. Para obter detalhes, consulte "Controles de site (arquivos de política)" na página 1089 e "Controles de autor (desenvolvedor)" na página 1093.

Arquivos SWF remotos não podem carregar nenhum recurso ou arquivo local.

Para obter mais informações sobre a segurança do Flash Player, consulte o tópico "Segurança" do Centro de desenvolvedores do Flash Player, em www.adobe.com/go/devnet_security_br.

Caixas de proteção locais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Arquivo local descreve qualquer arquivo que é referenciado usando o protocolo `file:` ou um caminho UNC (Convenção de nomenclatura universal). Arquivos SWF locais são colocados em uma de quatro caixas de proteção locais:

- A caixa de proteção local com sistema de arquivos — Por motivo de segurança, os aplicativos Flash Player e AIR colocam todos os arquivos locais na caixa de proteção local com sistema de arquivos (padrão). Nessa caixa de proteção, os códigos executáveis podem ler arquivos locais (usando a classe URLLoader, por exemplo), mas não podem se comunicar com a rede de nenhuma maneira. Isso garante ao usuário que os dados locais não podem ser vazados para a rede ou, de outra forma, compartilhados de maneira inadequada.

- A caixa de proteção local com a rede — ao compilar um arquivo SWF, é possível especificar que ele tem acesso à rede quando executado como um arquivo local (consulte "Configuração de tipo de caixa de proteção de arquivos SWF locais" na página 1084). Esses arquivos são colocados na caixa de proteção local com a rede. Arquivos SWF que são atribuídos à caixa de proteção local com a rede perdem o acesso a seus arquivos locais. Em troca, os arquivos SWF têm permissão para acessar dados da rede. No entanto um arquivo SWF local com a rede ainda não tem permissão para ler nenhum dado derivado da rede a menos que permissões estejam presentes para aquela ação, por meio de um arquivo de política de URL ou de uma chamada ao método `Security.allowDomain()`. Para conceder essa permissão, um arquivo de política de URL deve conceder permissão a *todos* os domínios usando `<allow-access-from domain="*"/>` ou usando `Security.allowDomain("*")`. Para obter mais informações, consulte "Controles de site (arquivos de política)" na página 1089 e "Controles de autor (desenvolvedor)" na página 1093.
- A caixa de proteção local confiável — arquivos SWF locais que estão registrados como confiáveis (pelos usuários ou pelos programas instaladores) são colocados na caixa de proteção local confiável. Administradores do sistema e usuários também têm a capacidade de reatribuir (mover) um arquivo SWF local para ou de uma caixa de proteção local confiável com base em considerações de segurança (consulte"Controles de administrador" na página 1086 e "Controles de usuário" na página 1088). Arquivos SWF atribuídos à caixa de proteção local confiável podem interagir com quaisquer outros arquivos SWF e carregar dados de qualquer lugar (remoto ou local).
- A caixa de proteção do aplicativo do AIR — esta caixa de proteção contém conteúdo instalado com o aplicativo do AIR em execução. Por padrão, o código executado na caixa de proteção do aplicativo do AIR podem realizar o cruzamento de scripts de códigos em qualquer domínio. No entanto, arquivos fora da caixa de proteção do aplicativo do AIR não têm permissão para executar o cruzamento de scripts de códigos na caixa de proteção do aplicativo. Por padrão, o código e o conteúdo na caixa de proteção do aplicativo do AIR podem carregar conteúdo e dados de qualquer domínio.

A comunicação entre as caixas de proteção local com rede e local com sistema de arquivos, bem como a comunicação entre as caixas de proteção remota e local com sistema de arquivos, é estritamente proibida. Permissão para essa comunicação não pode ser concedida por um aplicativo executado no Flash Player ou por um usuário ou administrador.

Script em qualquer direção entre arquivos HTML locais e arquivos SWF locais, por exemplo, usando a classe ExternalInterface, exige que tanto os arquivos HTML quanto os arquivos SWF envolvidos estejam na caixa de proteção local confiável. Isso ocorre porque os modelos de segurança locais de navegadores diferem do modelo de segurança local do Flash Player.

Arquivos SWF na caixa de proteção local com rede não podem carregar arquivos SWF na caixa de proteção local com sistema de arquivos. Arquivos SWF na caixa de proteção local com sistema de arquivos não podem carregar arquivos SWF na caixa de proteção local com rede.

A caixa de proteção do aplicativo do AIR

Adobe AIR 1.0 e posterior

O aplicativo Adobe AIR acrescenta uma caixa de proteção a mais, chamada *application*, ao modelo de caixa de proteção de segurança do Flash Player. Os arquivos instalados como parte de um aplicativo do AIR são carregados na caixa de proteção do aplicativo. Qualquer outro arquivo carregado pelo aplicativo terá restrições de segurança correspondentes àquelas especificadas pelo modelo normal de segurança do Flash Player.

Quando um aplicativo for instalado, todos os arquivos incluídos em um pacote do AIR são instalados em um diretório do aplicativo no computador do usuário. Os desenvolvedores podem fazer referência a esse diretório no código por meio do esquema de URL `app:/` (consulte "Esquemas de URI" na página 840). Todos os arquivos na árvore de diretório do aplicativo são atribuídos à caixa de proteção do aplicativo quando o aplicativo é executado. O conteúdo na caixa de proteção do aplicativo é agraciado com os privilégios completos disponíveis para o aplicativo do AIR , incluindo interação com o sistema de arquivos local.

Muitos dos aplicativos AIR só usam esses arquivos instalados localmente para executar o aplicativo. No entanto, os aplicativos AIR não estão restritos apenas aos arquivos no diretório do aplicativo - eles podem carregar qualquer tipo de arquivo de qualquer fonte. Isso inclui arquivos locais do computador do usuário, bem como arquivos de fontes externas disponíveis, como aqueles em uma rede local ou na Internet. O tipo de arquivo não tem nenhum impacto nas restrições de segurança, os arquivos HTML carregados têm os mesmos privilégios de segurança dos arquivos SWF carregados da mesma fonte.

O conteúdo na caixa de proteção de segurança do aplicativo tem acesso as APIs do AIR que o conteúdo de outras caixas de proteção não pode usar. Por exemplo, a propriedade `air.NativeApplication.nativeApplication.applicationDescriptor`, que retorna o conteúdo do arquivo descritor do aplicativo para o aplicativo, está restrita ao conteúdo na caixa de proteção de segurança do aplicativo. Outro exemplo de API restrita é a classe FileStream, que contém métodos de leitura e gravação no sistema de arquivos local.

As APIs do ActionScript disponíveis apenas para o conteúdo na caixa de proteção de segurança do aplicativo são indicadas com o logotipo do AIR na *Referência do ActionScript 3.0 para a plataforma Adobe Flash*. Usar essas APIs em outras caixas de proteção faz com que o tempo de execução lance uma exceção SecurityError.

No conteúdo HTML (em um objeto HTMLLoader), todas as APIs JavaScript do AIR (aquelas que estão disponíveis através da propriedade `window.runtime` ou através do objeto `air` durante o uso do arquivo AIRAliases.js) estão disponíveis para o conteúdo na caixa de proteção de segurança do aplicativo. O conteúdo HTML de outra caixa de proteção não tem acesso à propriedade `window.runtime`, portanto, esse conteúdo não pode acessar as APIs do AIR ou do Flash Player.

Os conteúdos executados na caixa de proteção do aplicativo do AIR têm as seguintes restrições a mais:

- Para conteúdo HTML na caixa de proteção de segurança do aplicativo, há limitações de uso de APIs que possam transformar dinamicamente as strings em código executável, após o código ser carregado. Isso é para evitar que o aplicativo injete inadvertidamente (e execute) código de fontes "não aplicativo" (como domínios de rede potencialmente inseguros). Um exemplo é o uso da função `eval()`. Para obter detalhes, consulte "Restrições de código de conteúdo em caixas de proteção distintas" na página 1123.
- Para evitar possíveis ataques de phishing, as tags `img` de conteúdo HTML nos objetos TextField do ActionScript são ignoradas no conteúdo SWF na caixa de proteção de segurança do aplicativo.
- O conteúdo na caixa de proteção do aplicativo não pode usar o protocolo `asfunction` do conteúdo HTML nos campos de texto do ActionScript 2.0.
- O conteúdo SWF na caixa de proteção do aplicativo não pode usar o cache entre domínios, um recurso que foi adicionado à Atualização 3 do Flash Player 9. Esse recurso permite que o Flash Player persista em colocar em cache o conteúdo de componente da plataforma Adobe e reutilize-o no conteúdo SWF carregado sob demanda (eliminando a necessidade de recarregar o conteúdo várias vezes).

Restrições a JavaScript dentro do AIR

Adobe AIR 1.0 e posterior

Ao contrário do conteúdo na caixa de proteção de segurança do aplicativo, o conteúdo JavaScript em uma caixa de proteção de segurança que não seja de aplicativo *pode* chamar a função `eval()` para executar código gerado dinamicamente a qualquer momento. Entretanto, existem restrições a JavaScript executado em uma caixa de proteção de segurança que não seja de aplicativo dentro do AIR. Isso inclui:

- O código JavaScript em uma caixa de proteção "não aplicativo" não tem acesso ao objeto `window.runtime` e, portanto, esse código não pode executar APIs do AIR.
- Por padrão, o conteúdo em uma caixa de proteção de segurança "não aplicativo" não pode usar chamadas XMLHttpRequest para carregar dados de outros domínios diferentes do domínio que chama a solicitação. Entretanto, o código do aplicativo pode conceder permissão para que o conteúdo "não aplicativo" faça isso, definindo um atributo `allowCrossdomainXHR` no frame ou iframe que o contém. Para obter mais informações, consulte "Restrições de código de conteúdo em caixas de proteção distintas" na página 1123.
- Há restrições na chamada do método `window.open()` de JavaScript. Para obter detalhes, consulte "Restrições na chamada do método window.open() de JavaScript" na página 1127.
- O conteúdo HTML nas caixas de proteção de segurança remota (rede) pode carregar apenas conteúdo CSS, `frame`, `iframe` e `img` de domínios remotos (de URLs de rede).
- O conteúdo HTML nas caixas de proteção local com sistema de arquivos, local com rede ou local confiável só podem carregar conteúdo CSS, `frame`, `iframe` e `img` de caixas de proteção locais (e não de URLs de aplicativo ou de rede).

Para obter detalhes, consulte "Restrições de código de conteúdo em caixas de proteção distintas" na página 1123.

Configuração de tipo de caixa de proteção de arquivos SWF locais

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um usuário final ou o administrador de um computador pode especificar que um arquivo SWF local é confiável, permitindo que ele carregue dados de todos os domínios, tanto locais quanto de rede. Isso é especificado nos diretórios Global Flash Player Trust e User Flash Player Trust. Para obter mais informações, consulte "Controles de administrador" na página 1086 e "Controles de usuário" na página 1088.

Para obter mais informações sobre caixas de proteção locais, consulte "Caixas de proteção locais" na página 1081.

Adobe Flash Professional

É possível configurar um arquivo SWF para a caixa de proteção local com sistema de arquivos ou a caixa de proteção local com rede definindo-se as configurações de publicação do documento na Ferramenta de autoria.

Adobe Flex

É possível configurar um arquivo SWF para a caixa de proteção local com sistema de arquivos ou a caixa de proteção local com rede configurando o sinalizador `use-network` no compilador do Adobe Flex. Para obter mais informações, consulte "Sobre as opções do compilador do aplicativo" em *Criação e implantação de aplicativos do Adobe Flex 3*.

A propriedade Security.sandboxType

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um autor de um arquivo SWF pode usar a propriedade estática somente leitura `Security.sandboxType` para determinar o tipo de caixa de proteção à qual o aplicativo Flash Player ou AIR atribuiu o arquivo SWF. A classe Security inclui constantes que representam valores possíveis da propriedade `Security.sandboxType`, da seguinte maneira:

- `Security.REMOTE` — o arquivo SWF é proveniente de uma URL da Internet e opera de acordo com as regras da caixa de proteção com base em domínio.
- `Security.LOCAL_WITH_FILE` — o arquivo SWF é um arquivo local, mas não foi considerado como confiável pelo usuário e não foi publicado com uma designação de rede. O arquivo SWF pode ler de fontes de dados locais, mas não pode se comunicar com a Internet.
- `Security.LOCAL_WITH_NETWORK` — o arquivo SWF é um arquivo local e não foi considerado como confiável pelo usuário, mas foi publicado com uma designação de rede. O arquivo SWF pode se comunicar com a Internet mas não pode ler de fontes de dados locais.
- `Security.LOCAL_TRUSTED` — o arquivo SWF é um arquivo local e foi considerado como confiável pelo usuário, seja usando o Gerenciador de configurações ou um arquivo de configuração de confiança do Flash Player. O arquivo SWF pode ler de fontes de dados locais e se comunicar com a Internet.
- `Security.APPLICATION` — o arquivo SWF está em execução em um aplicativo do AIR e foi instalado com o pacote (arquivo AIR) daquele aplicativo. Por padrão, os arquivos na caixa de proteção do aplicativo do AIR podem executar cross-script de qualquer arquivo em qualquer domínio. No entanto, arquivos fora da caixa de proteção do aplicativo do AIR não têm permissão para executar cross-script no arquivo AIR. Por padrão, os arquivos na caixa de proteção do aplicativo do AIR podem carregar conteúdo e dados de qualquer domínio.

Controles de permissão

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O modelo de segurança de tempo de execução do cliente Flash Player foi designado em torno de recursos que são objetos, como arquivos SWF, dados locais e URLs da Internet. *Participantes* são as partes que são as proprietárias ou que usam esses recursos. Os participantes podem exercer controles (configurações de segurança) sobre seus próprios recursos e cada recurso tem quatro participantes. O Flash Player reforça estritamente uma hierarquia de autoridade para esses controles, conforme mostrado na ilustração a seguir:

Hierarquia de controles de segurança

Isso significa, por exemplo, que se um administrador restringir o acesso a um recurso, nenhum outro participante poderá substituir essa restrição.

Nos aplicativos AIR, esses controles de permissão valem apenas para os conteúdos executados fora da caixa de proteção do aplicativo do AIR.

Controles de administrador

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um usuário administrativo de um computador (um usuário que fez logon com direitos administrativos) pode aplicar configurações de segurança do Flash Player que afetam todos os usuários do computador. Em um ambiente não empresarial, como em um computador doméstico, normalmente há um usuário que também tem acesso administrativo. Mesmo em um ambiente empresarial, usuários individuais podem ter direitos administrativos no computador.

Há dois tipos de controles de usuário administrativo:

- O arquivo mms.cfg
- O diretório Global Flash Player Trust

O arquivo mms.cfg

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O arquivo mms.cfg é um arquivo de texto que permite que os administradores ativem ou restrinjam o acesso a vários recursos. Quando o Flash Player é iniciado, lê as configurações de segurança desse arquivo e as usa para limitar a funcionalidade. O arquivo mms.cfg inclui configurações que o administrador usa para gerenciar recursos, como controles de privacidade, segurança de arquivo local, conexões de soquete, entre outros.

Um arquivo SWF pode acessar algumas informações sobre capacidades que foram desativadas chamando as propriedades `Capabilities.avHardwareDisable` e `Capabilities.localFileReadDisable`. No entanto a maioria das configurações no arquivo mms.cfg não podem ser consultadas no ActionScript.

Para reforçar as políticas de segurança e de privacidade independentemente do aplicativo para um computador, o arquivo mms.cfg deve ser modificado apenas por administradores do sistema. O arquivo mms.cfg não deve ser usado por instaladores de aplicativos. Embora um instalador que executa com privilégios administrativos possa modificar o conteúdo do arquivo mms.cfg, a Adobe considera esse uso como uma violação da confiança do usuário e recomenda que os criadores de instaladores nunca modifiquem o arquivo mms.cfg.

O arquivo mms.cfg é armazenado no seguinte local:

- Windows: *system*\Macromed\Flash\mms.cfg

  (por exemplo, C:\WINDOWS\system32\Macromed\Flash\mms.cfg)

- Mac: *app support*/Macromedia/mms.cfg

  (por exemplo, /Library/Application Support/Macromedia/mms.cfg)

Para obter mais informações sobre o arquivo mms.cfg, consulte o Guia de administração do Flash Player, em www.adobe.com/go/flash_player_admin_br.

O diretório Global Flash Player Trust

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Usuários administrativos e aplicativos de instalador podem registrar arquivos SWF locais especificados como confiáveis para todos os usuários. Esses arquivos SWF são atribuídos à caixa de proteção local confiável. Eles podem interagir com quaisquer outros arquivos SWF e podem carregar dados de qualquer lugar remoto ou local. Os arquivos são designados como confiáveis no diretório Global Flash Player Trust, no seguinte local:

- Windows: *system*\Macromed\Flash\FlashPlayerTrust

  (por exemplo, C:\WINDOWS\system32\Macromed\Flash\FlashPlayerTrust)

- Mac: *app support*/Macromedia/FlashPlayerTrust

  (por exemplo, /Library/Application Support/Macromedia/FlashPlayerTrust)

O diretório Flash Player Trust pode conter qualquer número de arquivos de texto, cada um dos quais lista caminhos confiáveis, com um caminho por linha. Cada caminho pode ser um arquivo SWF individual, um arquivo HTML ou um diretório. Linhas de comentários começam com o símbolo #. Por exemplo, um arquivo de configuração de confiança do Flash Player que contém o texto a seguir concede status confiável a todos os arquivos no diretório especificado e em todos os subdiretórios:

```
# Trust files in the following directories:
C:\Documents and Settings\All Users\Documents\SampleApp
```

Os caminhos listados em um arquivo de configuração de confiança devem sempre ser caminhos locais ou caminhos de rede SMB. Qualquer caminho HTTP em um arquivo de configuração confiável é ignorado. Apenas arquivos locais podem ser confiáveis.

Para evitar conflitos, dê a cada arquivo de configuração confiável um nome de arquivo correspondente ao aplicativo de instalação e use uma extensão de arquivo .cfg.

Como um desenvolvedor que distribui um arquivo SWF executado localmente por meio de um aplicativo instalador, é possível fazer com que o aplicativo instalador adicione um arquivo de configuração ao diretório Global Flash Player Trust, concedendo privilégios totais ao arquivo que você está distribuindo. O aplicativo instalador deve ser executado por um usuário com direitos administrativos. Ao contrário do arquivo ms.cfg, o diretório Global Flash Player Trust é incluído com o objetivo de aplicativos instaladores concederem permissões de confiança. Os usuários administrativos e os aplicativos instaladores podem designar aplicativos locais confiáveis usando o diretório Global Flash Player Trust.

Também há diretórios Flash Player Trust para usuários individuais (consulte "Controles de usuário" na página 1088).

Controles de usuário

Flash Player 9 e posterior

O Flash Player fornece três mecanismos diferentes em nível de usuário para definir permissões: a UI de Configurações, o Gerenciador de configurações e o diretório User Flash Player Trust.

A UI de Configurações e o Gerenciador de configurações

Flash Player 9 e posterior

A UI de Configurações é um mecanismo rápido e interativo para definir as configurações de um domínio específico. O Gerenciador de configurações tem uma interface mais detalhada e permite fazer alterações globais que afetam as permissões de muitos ou de todos os domínios. Além disso, quando um arquivo SWG solicita uma nova permissão, exigindo a tomada de decisões relativas a segurança ou privacidade durante a execução, são exibidas caixas de diálogo nas quais os usuários podem ajustar algumas configurações do Flash Player.

O Gerenciador de Configurações e a UI de Configurações oferecem opções relacionadas a segurança, como configurações de câmera e microfone, configurações de armazenamento de objetos compartilhados, configurações relacionadas a conteúdo pré-existente etc. Nem o Gerenciador de Configurações, nem a UI de Configurações estão disponíveis para os aplicativos AIR.

Nota: *Qualquer configuração feita no arquivo mms.cfg (consulte "Controles de administrador" na página 1086) não é refletida no Gerenciador de configurações.*

Para obter detalhes sobre o Gerenciador de configurações, consulte www.adobe.com/go/settingsmanager_br.

O diretório User Flash Player Trust

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Usuários e aplicativos instaladores podem registrar arquivos SWF locais como confiáveis. Esses arquivos SWF são atribuídos à caixa de proteção local confiável. Eles podem interagir com quaisquer outros arquivos SWF e podem carregar dados de qualquer lugar remoto ou local. Um usuário designa um arquivo como confiável no diretório User Flash Player Trust, que está no mesmo diretório que a área de armazenamento de objetos compartilhados, nos seguintes locais (os locais são específicos do usuário atual):

- Windows: app data\Macromedia\Flash Player\#Security\FlashPlayerTrust

  (por exemplo, C:\Documents and Settings\JohnD\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust no Windows XP ou C:\Users\JohnD\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust no Windows Vista)

  No Windows, a pasta Application Data fica oculta, por padrão. Para mostrar pastas e arquivos ocultos, selecione Meu computador para abrir o Windows Explorer, selecione Ferramentas > Opções de pasta e selecione a aba Exibir. Na guia Exibir, selecione o botão de opção Mostrar pastas e arquivos ocultos.

- Mac: app data/Macromedia/Flash Player/#Security/FlashPlayerTrust

  (por exemplo, /Users/JohnD/Library/Preferences/Macromedia/Flash Player/#Security/FlashPlayerTrust)

  Essas configurações afetam apenas o usuário atual, não outros usuários que fazem logon no computador. Se um usuário sem direitos administrativos instalar um aplicativo em sua própria parte do sistema, o diretório User Flash Player Trust permitirá que o instalador registre o aplicativo como confiável para aquele usuário.

Como um desenvolvedor que distribui um arquivo SWF executado localmente por meio de um aplicativo instalador, você pode fazer com que o aplicativo instalador adicione um arquivo de configuração ao diretório User Flash Player Trust, concedendo privilégios totais ao arquivo que está distribuindo. Mesmo nessa situação, o arquivo do diretório User Flash Player Trust é considerado um controle de usuário, porque uma ação do usuário (instalação) o inicia.

Também existe um diretório Global Flash Player Trust, usado pelo usuário administrativo ou por instaladores para registrar um aplicativo para todos os usuários de um computador (consulte "Controles de administrador" na página 1086).

Controles de site (arquivos de política)

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para disponibilizar dados do servidor Web para arquivos SWF de outros domínios, crie um arquivo de política do servidor. Um *arquivo de política* é um arquivo XML colocado em um local específico no servidor.

arquivos de política afetam o acesso a vários ativos, incluindo os seguintes:

- Dados em bitmaps, sons e vídeos
- Carregamento de arquivos de texto e XML
- Importação de arquivos SWF de outros domínios de segurança no domínio de segurança do arquivo SWF que está sendo carregado
- Acesso a soquete e conexões de soquete XML

Objetos ActionScript instanciam dois tipos diferentes de conexões de servidor: conexões de servidor com base em documento e conexões de soquete. Os objetos do ActionScript, como Loader, Sound, URLLoader e URLStream instanciam conexões de servidor com base em documento e esses objetos carregam um arquivo de uma URL. Os objetos Socket e XMLSocket do ActionScript fazem conexões de soquete que funcionam com fluxos de dados, não com documentos carregados.

Como o Flash Player oferece suporte a dois tipos de conexões de servidor, há dois tipos de arquivos de política: arquivos de política de URL e arquivos de política de soquete.

- Conexões com base em documento exigem *arquivos de política de URL*. Esses arquivos permitem que o servidor indique que seus dados e documentos estão disponíveis para arquivos SWF servidos em determinados domínios ou em todos os domínios.
- Conexões de soquete exigem *arquivos de política de soquetes* que ativam a rede diretamente no nível inferior do soquete TCP usando as classes Socket e XMLSocket.

O Flash Player exige que arquivos de política sejam transmitidos usando o mesmo protocolo que a tentativa de conexão quer usar. Por exemplo, quando você coloca um arquivo de política no servidor HTTP, os arquivos SWF de outros domínios recebem permissão para carregar dados dele como um servidor HTTP. No entanto, se não fornecer um arquivo de política de soquete no mesmo servidor, você estará proibindo que arquivos SWF de outros domínios se conectem ao servidor no nível de soquete. Em outras palavras, o meio pelo qual um arquivo de política é recuperado deve corresponder ao meio de conexão.

O uso e a sintaxe do arquivo de política são discutidos brevemente no restante desta seção, pois eles se aplicam aos arquivos SWF publicados para o Flash Player 10. A implementação do arquivo de política é um pouco diferente em versões anteriores do Flash Player, uma vez que versões sucessivas reforçaram a segurança do Flash Player. Para obter informações mais detalhadas sobre arquivos de política, consulte o tópico "Alterações em arquivos de política do Flash Player 9" do Centro de desenvolvedores do Flash Player, em www.adobe.com/go/devnet_security_br.

O código executado na caixa de proteção do aplicativo do AIR não exige um arquivo de políticas para acessar dados em uma URL ou um soquete. O código em um aplicativo do AIR executado em uma caixa de proteção que não seja de um aplicativo exige um arquivo de políticas.

arquivos de política mestre

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Por padrão, primeiro o Flash Player (e o conteúdo AIR que não está na caixa de proteção do aplicativo do AIR) procura um arquivo de política de URL denominado `crossdomain.xml` no diretório raiz do servidor e procura um arquivo de política de soquete na porta 843. Um arquivo em um desses locais é chamado de *arquivo de política mestre*. No caso de conexões de soquete, o Flash Player também procura um arquivo de política de soquete na mesma porta que a conexão principal. No entanto, um arquivo de política encontrado naquela porta não é considerado um arquivo de política mestre.

Além de especificar permissões de acesso, o arquivo de política mestre também pode conter uma instrução *meta-policy*. Uma metapolítica especifica quais locais podem conter arquivos de política. A metapolítica padrão dos arquivos de política de URL é "somente mestre", o que significa que /crossdomain.xml é o único arquivo de política permitido no servidor. A metapolítica padrão para arquivos de política de soquete é "all", o que significa que qualquer soquete no host pode servir um arquivo de política de soquete.

Nota: *No Flash Player 9 e anterior, a metapolítica padrão de arquivos de política de URL era "all", o que significa que qualquer diretório pode conter um arquivo de política. Se você implantou aplicativos que carregam arquivos de política de locais diferentes do arquivo padrão /crossdomain.xml e se esses aplicativos estiverem executando agora no Flash Player 10, verifique se você (ou outro administrador do servidor) modificou o arquivo de política mestre para permitir arquivos de política adicionais. Para obter informações sobre como especificar uma metapolítica diferente, consulte o tópico "Alterações em arquivos de política do Flash Player 9" do Centro de desenvolvedores do Flash Player, em www.adobe.com/go/devnet_security_br.*

Um arquivo SWF pode verificar se existe um nome de arquivo de política diferente ou um local de diretório diferente chamando o método `Security.loadPolicyFile()`. No entanto, se o arquivo de política mestre não especificar que o local de destino pode servir arquivos de política, a chamada para `loadPolicyFile()` não terá nenhum efeito, mesmo que haja um arquivo de política naquele local. Chame `loadPolicyFile()` antes de tentar qualquer operação de rede que exija o arquivo de política. O Flash Player enfileira automaticamente solicitações de rede por trás de tentativas do arquivo de política correspondente. Portanto, é aceitável, por exemplo, chamar `Security.loadPolicyFile()` imediatamente antes de iniciar uma operação de rede.

Ao verificar um arquivo de política mestre, o Flash Player aguarda três segundos por uma resposta do servidor. Se uma resposta não for recebida, o Flash Player assumirá que não existe nenhum arquivo de política. No entanto não há nenhum valor de tempo limite padrão para chamadas para `loadPolicyFile()`. O Flash Player assume que o arquivo que está chamando existe, e aguarda quanto tempo for necessário para carregá-lo. Portanto, para verificar se um arquivo de política mestre está carregado, use `loadPolicyFile()` para chamá-lo explicitamente.

Mesmo que o método seja denominado `Security.loadPolicyFile()`, um arquivo de política não será carregado até que uma chamada de rede que exija um arquivo de política seja emitida. Chamadas para `loadPolicyFile()` simplesmente informam ao Flash Player onde procurar por arquivos de política quando eles são necessários.

Não é possível receber notificação de quando uma solicitação de arquivo de política é iniciada ou concluída e não há motivo para isso. O Flash Player executa verificações de políticas assincronamente e aguarda automaticamente para iniciar conexões até após o êxito das verificações do arquivo de política.

As seções a seguir contêm informações que se aplicam apenas a arquivos de política de URL. Para obter mais informações sobre arquivos de política de soquete, consulte "Conexão a soquetes" na página 1108.

Escopo do arquivo de política de URL

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O arquivo de política de URL se aplica apenas ao diretório do qual ele é carregado e a seus diretórios filho. Um arquivo de política no diretório raiz se aplica ao todo o servidor. Um arquivo de política carregado de um subdiretório arbitrário aplica-se apenas àquele diretório e a seus subdiretórios.

Um arquivo de política afeta o acesso somente ao servidor específico onde ele reside. Por exemplo, um arquivo de política localizado em https://www.adobe.com:8080/crossdomain.xml aplica-se apenas a chamadas de carregamento de dados feitas para www.adobe.com por HTTPS na porta 8080.

Especificação de permissões de acesso em um arquivo de política de URL

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Um arquivo de política contém uma única tag `<cross-domain-policy>` que, por sua vez, contém zero ou mais tags `<allow-access-from>`. Cada tag `<allow-access-from>` contém um atributo, `domain`, que especifica um endereço IP exato, um domínio exato ou um domínio curinga (qualquer domínio). Os domínios curinga são indicados de duas maneiras:

- Por um único asterisco (*), que corresponde a todos os domínios e a todos os endereços IP.
- Por um asterisco seguido por um sufixo, que corresponde apenas aos domínios que terminam com o sufixo especificado.

Sufixos devem começar com um ponto. No entanto domínios curinga com sufixos podem corresponder a domínios que consistem apenas no sufixo sem o ponto inicial. Por exemplo, xyz.com é considerado como parte de *.xyz.com. Curingas não são permitidos em especificações de domínios IP.

O exemplo a seguir mostra um arquivo de política de URL que permite acesso a arquivos SWF originados de *.example.com, www.friendOfExample.com e 192.0.34.166:

```
<?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>
```

Se você especificar um endereço IP, o acesso será concedido somente a arquivos SWF carregados daquele endereço IP usando a sintaxe IP (por exemplo, http://65.57.83.12/flashmovie.swf). O acesso não é concedido a arquivos SWF que usam a sintaxe de nome de domínio. O Flash Player não executa resolução de DNS.

É possível permitir acesso a documentos originários de qualquer domínio, conforme mostrado no exemplo a seguir:

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

Cada tag `<allow-access-from>` também tem o atributo opcional `secure` que é padronizado como `true`. Se o arquivo de política estiver em um servidor HTTPS e você quiser permitir que arquivos SWF em um servidor não HTTPS carreguem dados do servidor HTTPS, poderá definir o atributo como `false`.

A configuração do atributo `secure` como `false` pode comprometer a segurança oferecida pelo HTTPS. Especificamente, a configuração desse atributo como `false` abre conteúdo de segurança a ataques por falsificação. A Adobe recomenda enfaticamente que você não defina o atributo `secure` como `false`.

Se os dados a serem carregados estiverem em um servidor HTTPS, mas o carregamento do arquivo SWF estiver em um servidor HTTP, a Adobe recomenda mover o arquivo SWF que está sendo carregado para um servidor HTTPS. Fazer isso permite manter todas as cópias de seus dados seguras sob a proteção de HTTPS. No entanto, se você decidir que deve manter o arquivo SWF que está sendo carregado em um servidor HTTP, adicione o atributo `secure="false"` à tag `<allow-access-from>`, conforme mostrado no código a seguir:

```
<allow-access-from domain="www.example.com" secure="false" />
```

Outro elemento que pode ser usado para permitir acesso é a tag `allow-http-request-headers-from`. Esse elemento concede a um cliente, que hospeda conteúdo de outro domínio, permissão para enviar cabeçalhos definidos pelo usuário ao seu domínio. Enquanto a tag `<allow-access-from>` concede a outros domínios permissão para enviar dados ao seu domínio, a tag `allow-http-request-headers-from` concede a outros domínios permissão para enviar dados a seu domínio na forma de cabeçalhos. No exemplo a seguir, qualquer domínio tem permissão para enviar o cabeçalho SOAPAction ao domínio atual:

```
<cross-domain-policy>
    <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

Se a instrução `allow-http-request-headers-from` estiver no arquivo de política mestre, ela será aplicada a todos os diretórios no host. Caso contrário, ela será aplicada apenas ao diretório e subdiretórios do arquivo de política que contêm a instrução.

Pré-carregamento de arquivos de política

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O carregamento de dados de um servidor ou a conexão a um soquete é uma operação assíncrona. O Flash Player simplesmente aguarda que o download do arquivo de política seja concluído antes de iniciar a operação principal. No entanto, a extração de dados de pixels de imagens ou a extração de dados de amostra de sons é uma operação síncrona. O arquivo de política deve ser carregado para que os dados possam ser extraídos. Ao carregar a mídia, especifique que ela verifique a existência de um arquivo de política:

- Ao usar o método `Loader.load()`, defina a propriedade `checkPolicyFile` do parâmetro `context`, que é um objeto LoaderContext.
- Ao incorporar uma imagem em um arquivo de texto usando a tag `<img>`, defina o atributo `checkPolicyFile` da tag `<img>` como `"true"`, da seguinte maneira:

  ```
  <img checkPolicyFile = "true" src = "example.jpg">
  ```

- Ao usar o método `Sound.load()`, defina a propriedade `checkPolicyFile` do parâmetro `context`, que é um objeto SoundLoaderContext.
- Ao usar a classe NetStream, defina a propriedade `checkPolicyFile` do objeto NetStream.

Ao definir um desses parâmetros, o Flash Player primeiro verifica se há qualquer arquivo de política que já foi baixado naquele domínio. Em seguida, ele procura o arquivo de política no local padrão do servidor, verificando as instruções `<allow-access-from>` e a presença de uma metapolítica. Finalmente, ele considera quaisquer chamadas pendentes para o método `Security.loadPolicyFile()` para verificar se elas estão em escopo.

Controles de autor (desenvolvedor)

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A API ActionScript principal usada para conceder privilégios de segurança é o método `Security.allowDomain()` que concede privilégios a arquivos SWF nos domínios especificados por você. No exemplo a seguir, um arquivo SWF concede acesso a arquivos SWF servidos no domínio www.example.com:

```
Security.allowDomain("www.example.com")
```

Esse método concede permissões para o seguinte:

- Entre scritps entre arquivos SWF (consulte "Cross-scripting" na página 1101)
- Acesso à lista de exibição (consulte "Como percorrer a lista de exibição" na página 1104)
- Detecção de eventos (consulte "Segurança de eventos" na página 1104)
- Acesso total à propriedades e métodos do objeto Stage (consulte"segurança de Palco" na página 1103)

O objetivo principal da chamada do método `Security.allowDomain()` é conceder permissão para arquivos SWF em um domínio externo para executar script do arquivo SWF chamando o método `Security.allowDomain()`. Para obter mais informações, consulte "Cross-scripting" na página 1101.

Especificar um endereço IP como parâmetro do método `Security.allowDomain()` não permite acesso de todas as partes que se originam no endereço IP especificado. Ao contrário, isso permite o acesso apenas de uma parte que contém o endereço IP especificado como sua URL, em vez de um nome de domínio mapeia para o endereço IP. Por exemplo, se o nome do domínio www.example.com for mapeado para o endereço IP 192.0.34.166, uma chamada para `Security.allowDomain("192.0.34.166")` não concederá acesso a www.example.com.

É possível passar o curinga `"*"` para o método `Security.allowDomain()` para permitir acesso a todos os domínios. Como ele concede permissão para arquivos SWF de *todos* os domínios para executarem script na chamada do arquivo, use o curinga `"*"` com cuidado.

O ActionScript inclui uma segunda API de permissão, chamada `Security.allowInsecureDomain()`. Esse método faz a mesma coisa que o método `Security.allowDomain()`, exceto que, quando chamado de um arquivo SWF servido por uma conexão HTTPS segura, ele permite adicionalmente acesso ao arquivo SWF de chamada por outros arquivos SWF que são servidos em um protocolo inseguro, como HTTP. No entanto, não é uma boa prática de segurança permitir script entre arquivos de um protocolo seguro (HTTPS) e arquivos de protocolos inseguros (como HTTP). Isso pode abrir conteúdo seguro a ataques de falsificação. Esses ataques podem ocorrer da seguinte maneira: como o método `Security.allowInsecureDomain()` permite acesso aos dados de HTTPS seguro por arquivos SWF servidos em conexões HTTP, um invasor situado entre o servidor HTTP e seus usuários pode substituir seu arquivo SWF de HTTP por um arquivo próprio, que pode então acessar seus dados HTTPS.

Importante: *O código executado na caixa de proteção do aplicativo do AIR não tem permissão para chamar os métodos* `allowDomain()` *ou* `allowInsecureDomain()` *da classe de Segurança.*

Outro método importante relacionado à segurança é o método `Security.loadPolicyFile()` que faz com que o Flash Player verifique se há um arquivo de política em um local não padrão. Para obter mais informações, consulte "Controles de site (arquivos de política)" na página 1089.

Restrição de APIs de rede

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

APIs de rede podem ser restringidas de duas maneiras. Para impedir atividades mal-intencionadas, o acesso a portas comumente reservadas é bloqueado. Você não pode substituir esses bloqueios em seu código. Para controlar o acesso de um arquivo SWF à funcionalidade da rede com relação a outras portas, é possível usar a configuração `allowNetworking`.

Portas bloqueadas

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Player e o Adobe AIR têm restrições sobre o acesso HTTP a determinadas portas, assim como navegadores. Solicitações HTTP não são permitidas para determinadas portas padrão que convencionalmente são usadas para tipos de servidores não HTTP.

Qualquer API que acesse uma URL da rede está sujeita a essas restrições de bloqueio de porta. A única exceção são APIs que chamam soquetes diretamente, como `Socket.connect()` e`XMLSocket.connect()`, ou chamadas para `Security.loadPolicyFile()` nas quais um arquivo de política de soquete está sendo carregado. Conexões de soquete são permitidas ou negadas por meio do uso de arquivos de política de soquete no servidor de destino.

A lista a seguir mostra as APIs do ActionScript 3.0 às quais se aplica o bloqueio de portas:

`FileReference.download()`, `FileReference.upload()`, `Loader.load()`, `Loader.loadBytes()`, `navigateToURL()`, `NetConnection.call()`, `NetConnection.connect()`, `NetStream.play()`, `Security.loadPolicyFile()`, `sendToURL()`, `Sound.load()`, `URLLoader.load()`, `URLStream.load()`

O bloqueio de portas também se aplica à importação da Biblioteca compartilhada, ao uso da tag `<img>` em campos de texto e ao carregamento de arquivos SWF em uma página HTML usando as tags `<object>` e`<embed>`.

O bloqueio de portas também se aplica ao uso da tag `<img>` em campos de texto e ao carregamento de arquivos SWF em uma página HTML usando as tags `<object>` e`<embed>`.

As listas a seguir mostram quais portas são bloqueadas:

HTTP: 20 (dados ftp), 21 (controle ftp)

HTTP e 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)

Uso do parâmetro allowNetworking

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível controlar o acesso de um arquivo SWF à funcionalidade da rede definindo o parâmetro `allowNetworking` nas tags `<object>` e`<embed>` na página HTML que contém o conteúdo do SWF.

Os valores possíveis de `allowNetworking` são:

- `"all"` (o padrão) — todas as APIs de rede têm permissão no arquivo SWF.
- `"internal"` — o arquivo SWF não pode chamar APIs de interação de navegador ou navegação de navegador, listadas posteriormente nesta seção, mas ele pode chamar quaisquer outras APIs de rede.
- `"none"` — o arquivo SWF não pode chamar APIs de interação de navegador ou de navegação de navegador, listadas posteriormente nesta seção, e não pode usar quaisquer APIs de comunicação SWF com SWF, também listadas posteriormente.

O parâmetro `allowNetworking` foi projetado para uso principalmente quando o arquivo SWF e a página HTML que o delimita estão em domínios diferentes. O uso do valor de `"internal"` ou de `"none"` não é recomendado quando o arquivo SWF que está sendo carregado é do mesmo domínio que as páginas HTML que o delimitam, porque não é possível garantir que um arquivo SWF sempre seja carregado com a página HTML pretendida. Partes não confiáveis podem carregar um arquivo SWF do seu domínio sem HTML delimitado, em cujo caso a restrição `allowNetworking` não funcionará como pretendido.

A chamada de uma API impedida lança uma exceção SecurityError.

Adicione o parâmetro `allowNetworking` e defina seu valor nas tags `<object>` e `<embed>` da página HTML que contém uma referência ao arquivo SWF, conforme mostrado neste exemplo:

```
<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>
```

Uma página HTML também pode usar um script para gerar tags de incorporação SWF. É necessário alterar o script para que ele insira as configurações corretas de `allowNetworking`. As páginas HTML geradas pelo Adobe Flash Professional e pelo Adobe Flash Builder usam a função `AC_FL_RunContent()` para incorporar referências aos arquivos SWF. Adicione as configurações do parâmetro `allowNetworking` ao script, como no seguinte:

```
AC_FL_RunContent( ... "allowNetworking", "none", ...)
```

As seguintes APIs são impedidas quando `allowNetworking` está definido como `"internal"`:

`navigateToURL()`, `fscommand()`, `ExternalInterface.call()`

Além das APIs da lista anterior, as seguintes APIs também são impedidas quando o parâmetro `allowNetworking` está definido como `"none"`:

`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()`

Mesmo que a configuração `allowNetworking` selecionada permita que um arquivo SWF use uma API de rede, pode haver outras restrições com base nas limitações da caixa de proteção de segurança (consulte "Caixas de proteção de segurança" na página 1081).

Quando `allowNetworking` está definido como `"none"`, não é possível fazer referência a mídia externa em uma tag `<img>` na propriedade `htmlText` de um objeto TextField (uma exceção SecurityError é lançada).

Quando `allowNetworking` está definido como `"none"`, um símbolo de uma biblioteca compartilhada importada adicionado no Flash Professional (não no ActionScript) é bloqueado em tempo de execução.

Segurança de modo de tela cheia

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Player 9.0.27.0 e versões posteriores oferecem suporte ao modo de tela cheia, no qual o conteúdo executado no Flash Player pode ocupar toda a tela. Para entrar em modo de tela cheia, a propriedade `displayState` do Palco é definida como a constante `StageDisplayState.FULL_SCREEN`. Para obter mais informações, consulte "Trabalho com o modo de tela cheia" na página 169.

Existem mais considerações sobre segurança para arquivos SWF executados em uma área de segurança remota.

Para ativar o modo de tela cheia, nas tags `<object>` e `<embed>` na página HTML que contém uma referência ao arquivo SWF, adicione o parâmetro `allowFullScreen` com o valor definido como `"true"` (o valor padrão é `"false"`), conforme mostrado no exemplo a seguir:

```
<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>
```

Uma página HTML também pode usar um script para gerar tags de incorporação SWF. É necessário alterar o script para que ele insira as configurações corretas de `allowFullScreen`. As páginas HTML geradas pelo Flash Professional e pelo Flash Builder usam a função `AC_FL_RunContent()` para incorporar referências a arquivos SWF e você precisa adicionar as configurações do parâmetro `allowFullScreen`, conforme a seguir:

```
AC_FL_RunContent( ... "allowFullScreen", "true", ...)
```

O ActionScript que inicia o modo de tela cheia pode ser chamado apenas em resposta a um evento do mouse ou do teclado. Se ele for chamado em outras situações, o Flash Player lançará uma exceção.

Um mensagem é exibida quando o conteúdo entra em modo de tela cheia, fornecendo ao usuário instruções sobre como sair e retornar ao modo normal. A mensagem é exibida por alguns segundos e, em seguida, desaparece.

No caso de conteúdo executado em um navegador, o uso do teclado fica restrito no modo de tela cheia. No Flash Player 9, só é dado suporte para atalhos de teclado que retornam o aplicativo ao modo normal, como pressionar a tecla Escape. Os usuários não podem inserir texto em campos de texto ou navegar pela tela. No Flash Player 10 e em versões posteriores, há suporte para certas teclas que não são impressas (especificamente as teclas de seta, espaço e Tab). No entanto, a entrada de texto ainda não é permitida.

O modo de tela cheia sempre é permitido no player independente ou em um arquivo de projetor. Além disso, o uso do teclado (inclusive para entrada de texto) é totalmente suportado nesses ambientes.

Chamar a propriedade `displayState` de um objeto Stage lança uma exceção para qualquer chamador que não esteja na mesma caixa de proteção de segurança que o proprietário do Palco (o arquivo SWF principal). Para obter mais informações, consulte "segurança de Palco" na página 1103.

Os administradores podem desativar o modo de tela cheia de arquivos SWF em execução em navegadores configurando `FullScreenDisable = 1` no arquivo mms.cfg. Para obter detalhes, consulte "Controles de administrador" na página 1086.

Em um navegador, um arquivo SWF deve ser contido em uma página HTML para permitir o modo de tela cheia.

Segurança do modo interativo de tela inteira

Flash Player 11.3 e posterior, Adobe AIR 1.0 e posterior

O Flash Player 11.3 e posterior suporta o modo interativo de tela inteira, no qual o conteúdo em execução no Flash Player pode preencher a tela inteira *e aceitar a entrada de texto*. Para entrar no modo interativo de tela inteira, a propriedade `displayState` do Estágio é definida como a constante `StageDisplayState.FULL_SCREEN_INTERACTIVE`. Para obter mais informações, consulte "Trabalho com o modo de tela cheia" na página 169.

Existem mais considerações sobre segurança para arquivos SWF executados em uma área de segurança remota.

Para ativar o modo de tela inteira, nas marcas de formatação `<object>` e `<embed>` da página HTML que contém referência ao arquivo SWF, adicione o parâmetro `allowFullScreenInteractive`, com o valor definido como `"true"` (o valor padrão é `"false"`), conforme mostra no exemplo a seguir:

```
<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>
```

Uma página HTML também pode usar um script para gerar tags de incorporação SWF. Você deve alterar o script para que insira as configurações `allowFullScreenInteractive` apropriadas. As páginas HTML geradas pelo Flash Professional e Flash Builder usam a função `AC_FL_RunContent()` para incorporar referências a arquivos SWF e você precisa adicionar as configurações de parâmetro `allowFullScreenInteractive`, como no seguinte:

```
AC_FL_RunContent( ... "allowFullScreenInteractive", "true", ...)
```

O ActionScript que inicia o modo interativo de tela inteira pode ser chamado somente em resposta a um evento de mouse ou teclado. Se ele for chamado em outras situações, o Flash Player lançará uma exceção.

Uma mensagem de sobreposição aparece quando o conteúdo entra no modo interativo de tela inteira. A mensagem exibe o domínio da página de tela inteira, instruções sobre como sair do modo de tela inteira e um botão **Permitir**. A sobreposição persiste até o usuário clicar em**Permitir**, reconhecendo que está no modo interativo de tela inteira.

Os administradores pode desativar o modo interativo de tela inteira para arquivos SWF em execução nos navegadores ao configurar `FullScreenInteractiveDisable = 1` no arquivo mms.cfg. Para obter detalhes, consulte "Controles de administrador" na página 1086.

Em um navegador, um arquivo SWF deve estar contido em uma página HTML para permitir o modo interativo de tela inteira.

Carregamento de conteúdo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O conteúdo do Flash Player e do AIR pode carregar muitos outros tipos de conteúdos, entre eles o seguinte:

- Arquivos SWF
- Imagens
- Som
- Vídeo
- Arquivos HTML (apenas AIR)
- JavaScript (apenas AIR)

Carregamento de arquivos SWF e imagens com a classe Loader

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Use a classe Loader para carregar arquivos SWF e imagens (arquivos JPG, GIF ou PNG). Qualquer arquivo SWF, a não ser um que esteja na caixa de proteção local com arquivos do sistema, pode carregar arquivos SWF e imagens de qualquer domínio de rede. Apenas arquivos SWF em caixas de proteção locais podem carregar arquivos SWF e imagens do sistema de arquivos local. No entanto arquivos na caixa de proteção local com rede podem carregar apenas arquivos SWF locais que estejam na mesma caixa de proteção confiável local ou local com rede. Os arquivos SWF na caixa de proteção local com rede carregam conteúdo local que não sejam arquivos SWF (como imagens), no entanto eles não podem acessar dados no conteúdo carregado.

Ao carregar um arquivo SWF a partir de uma origem não confiável (como um domínio diferente daquele do arquivo SWF da raiz do objeto Loader), convém definir uma máscara para o objeto Loader para impedir que o conteúdo carregado (que é filho do objeto Loader) seja desenhado em partes do Palco fora daquela máscara, conforme mostrado no código a seguir:

```
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);
```

Ao chamar o método `load()` do objeto Loader, é possível especificar um parâmetro `context` que é um objeto LoaderContext. A classe LoaderContext inclui três propriedades que permitem definir o contexto de como o conteúdo carregado pode ser usado:

- `checkPolicyFile`: Use essa propriedade apenas ao carregar um arquivo de imagem (não um arquivo SWF). Especifique-a para um arquivo de imagem de um domínio diferente daquele do arquivo que contém o objeto Loader. Se você definir a propriedade como `true`, o Loader verificará o servidor de origem para obter um arquivo de política de URL (consulte"Controles de site (arquivos de política)" na página 1089). Se o servidor conceder permissão ao domínio Loader, o ActionScript de arquivos SWF no domínio Loader poderá acessar dados na imagem carregada. Em outras palavras, você pode usar a propriedade `Loader.content` para obter uma referência ao objeto Bitmap que representa a imagem carregada ou os métodos `BitmapData.draw()` ou `BitmapData.drawWithQuality()` para acessar pixels da imagem carregada. O método `drawWithQuality` está disponível no Flash Player 11.3 e superior; AIR 3.3 e superior.
- `securityDomain`: Só use essa propriedade ao carregar um arquivo SWF (não uma imagem). Especifique-a para um arquivo SWF de um domínio diferente daquele do arquivo que contém o objeto Loader. Apenas dois valores são suportados atualmente para a propriedade `securityDomain`: `null` (o padrão) e `SecurityDomain.currentDomain`. Se você especificar `SecurityDomain.currentDomain`, ele solicitará que o arquivo SWF carregado seja *importado* para a caixa de proteção do arquivo SWF que está sendo carregado, indicando que ele funcionará como se fosse carregado do próprio servidor do arquivo SWF que está sendo carregado. Isso será permitido apenas se um arquivo de política de URL for encontrado no servidor do arquivo SWF carregado, permitindo acesso pelo domínio do arquivo SWF que está sendo carregado. Se o arquivo de política for encontrado, o carregador e o carregado poderão executar script livremente um no outro assim que o carregamento for iniciado, uma vez que estão na mesma caixa de proteção. Observe que a importação da caixa de proteção pode ser substituída principalmente executando um carregamento comum e, em seguida, fazendo com que o arquivo SWF chame o método `Security.allowDomain()`. Esse último método pode ser mais fácil de usar, pois o arquivo SWF carregado estará em sua própria caixa de proteção natural e, portanto, capaz de acessar recursos de seu próprio servidor real.
- `applicationDomain`: Use essa propriedade apenas ao carregar um arquivo SWF gravado no ActionScript 3.0 (não em uma imagem ou arquivo SWF gravado no ActionScript 1.0 ou no ActionScript 2.0). Ao carregar o arquivo, é possível especificar se ele será colocado em um domínio de aplicativo específico, em vez de ser colocado como padrão em um novo domínio de aplicativo que é filho do domínio do aplicativo do arquivo SWF que está sendo carregado. Observe que os domínios de aplicativos são subunidades de domínios de segurança e portanto você poderá especificar um domínio de aplicativo de destino apenas se o arquivo SWF que está sendo carregado for de seu próprio domínio de segurança, seja porque ele é de seu próprio servidor ou porque você o importou com êxito em seu domínio de segurança usando a propriedade `securityDomain`. Se você especificar um domínio de aplicativo, mas o arquivo SWF carregado fizer parte de um domínio de segurança diferente, o domínio especificado em `applicationDomain` será ignorado. Para obter mais informações, consulte "Trabalhar com domínios de aplicativo" na página 149.

Para obter detalhes, consulte "Especificação do contexto do carregamento" na página 204.

Uma propriedade importante de um objeto Loader é a propriedade `contentLoaderInfo` que é um objeto LoaderInfo. Ao contrário da maioria dos outros objetos, um objeto LoaderInfo é compartilhado entre o arquivo SWF que está sendo carregado e o conteúdo carregado e sempre está acessível para as duas partes. Quando o conteúdo carregado é um arquivo SWF, ele pode acessar o objeto LoaderInfo por meio da propriedade `DisplayObject.loaderInfo`. Objetos LoaderInfo incluem informações, como o progresso do carregamento, as URLs do carregador e do carregado, o relacionamento de confiança entre o carregador e o carregado e outras informações. Para obter mais informações, consulte "Monitoramento do progresso do carregamento" na página 203.

Carregamento de som e vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Todos o conteúdo, além dos que estão na área de proteção local com o sistema de arquivos, têm permissão para carregar som e vídeo de origens da rede, usando os métodos `Sound.load()`, `NetConnection.connect()` e `NetStream.play()` .

Somente o conteúdo local com o sistema de arquivos e aplicativos AIR podem carregar mídia do sistema de arquivos local. Somente o conteúdo na área de segurança local com o sistema de arquivos, a área de segurança do aplicativo do AIR ou a área de segurança local confiável podem acessar dados nos arquivos carregados.

Há outras restrições ao acessar dados de mídia carregada. Para obter detalhes, consulte "Acesso à mídia carregada como dados" na página 1105.

Carregamento de arquivos SWF e de imagens usando a tag <img> em um campo de texto

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível carregar arquivos SWF e bitmaps em um campo de texto usando a tag `<img>`, conforme mostrado no código a seguir:

```
<img src = 'filename.jpg' id = 'instanceName' >
```

É possível carregar conteúdo carregado dessa maneira usando o método `getImageReference()` da ocorrência TextField, conforme mostrado no código a seguir:

```
var loadedObject:DisplayObject = myTextField.getImageReference('instanceName');
```

No entanto observe que arquivos SWF e imagens carregados dessa maneira são colocados na caixa de proteção que corresponde a sua origem.

Ao carregar um arquivo de imagem usando uma tag `<img>` em um campo de texto, o acesso aos dados da imagem pode ser permitido por um arquivo de política de URL. É possível verificar se há um arquivo de política adicionando um atributo `checkPolicyFile` à tag `<img>`, como no código a seguir:

```
<img src = 'filename.jpg' checkPolicyFile = 'true' id = 'instanceName' >
```

Ao carregar um SWF usando uma tag `<img>` em um campo de texto, é possível permitir acesso aos dados daquele arquivo SWF por meio de uma chamada ao método `Security.allowDomain()`.

Ao usar uma tag `<img>` em um campo de texto para carregar um arquivo externo (em vez de usar uma classe Bitmap incorporada em seu SWF), um objeto Loader é criado automaticamente como um filho do objeto TextField, e o arquivo externo é carregado naquele Loader exatamente como se você tivesse usado um objeto Loader no ActionScript para carregar o arquivo. Nesse caso, o método `getImageReference()` retorna o Loader que foi criado automaticamente. Nenhuma verificação de segurança é necessária para acessar esse objeto Loader porque ele está na mesma caixa de proteção de segurança que o objeto de chamada.

No entanto quando você faz referência à propriedade `content` do objeto Loader para acessar a mídia carregada, regras de segurança são aplicadas. Se o conteúdo for uma imagem, será necessário implementar um arquivo de política de URL e, se o conteúdo for um arquivo SWF, você precisará fazer com que o código no arquivo SWF chame o método `allowDomain()`.

Adobe AIR

Na caixa de proteção do aplicativo, as marcas <img> em um campo de texto são ignoradas para evitar ataques de phishing. Além disso, o código executado na caixa de proteção do aplicativo não pode chamar o método de Segurança `allowDomain()`.

Conteúdo entregue usando servidores RMTP

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Media Server usa o protocolo RTMP para servir dados, áudio e vídeo. É possível carregar esta mídia usando o método `connect()` da classe NetConnection, passando uma URL de RTMP como o parâmetro. O Flash Media Server pode restringir conexões e impedir que conteúdo seja baixado, com base no domínio do arquivo solicitante. Para obter detalhes, consulte a documentação on-line do Flash Media Server em www.adobe.com/go/learn_fms_docs_br.

Para usar os métodos `BitmapData.draw()`,`BitmapData.drawWithQuality()` e `SoundMixer.computeSpectrum()` e extrair gráficos de tempo de execução e dados de som dos fluxos RTMP, você deve permitir o acesso no servidor. Utilize o ActionScript de lado do servidor `Client.videoSampleAccess` e as propriedades `Client.audioSampleAccess` para permitir acesso a diretórios específicos no Flash Media Server. Para obter mais informações, consulte a Referência de linguagem do ActionScript de lado do servidor. (O método `drawWithQuality` está disponível no Flash Player 11.3 e superior; AIR 3.3 e superior.)

Cross-scripting

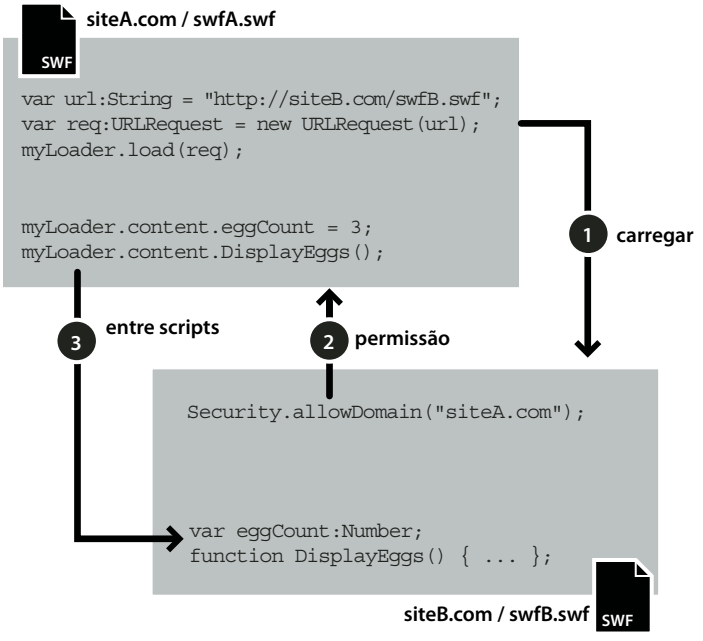
Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Se dois arquivos escritos com o ActionScript 3.0 , ou dois arquivos HTML executados no AIR, forem servidos no mesmo domínio, por exemplo, se a URL de um arquivo SWF for http://www.example.com/swfA.swf e a URL do outro for http://www.example.com/swfB.swf, o código definido em um arquivo poderá examinar e modificar variáveis, objetos, propriedades, métodos e assim por diante na ordem e vice-versa. Isso é chamado de *cross-scripting*.

Se dois arquivos forem servidos em domínios diferentes, por exemplo, http://siteA.com/swfA.swf e http://siteB.com/siteB.swf, por padrão, o Flash Player e o AIR não permitirão que o swfA.swf execute script do swfB.swf nem que o swfB.swf execute script do swfA.swf. O arquivo SWF dá permissão a arquivos SWF de outros domínios chamando `Security.allowDomain()`. Com a chamada de `Security.allowDomain("siteA.com")`, o swfB.swf concede aos arquivos SWF do siteA.swf permissão para executar script nele.

O cross-scripting não é suportado entre arquivos SWF AVM1 e SWF AVM2. Um arquivo SWF AVM1 é criado usando o ActionScript 1.0 ou o ActionScript 2.0. (AVM1 e AVM2 se referem a ActionScript Virtual Machine.) No entanto é possível usar a classe LocalConnection para enviar dados entre o AVM1 e o AVM2.

Em qualquer situação entre domínios, é importante que as duas partes envolvidas sejam bem-definidas. Para fins desta discussão, o lado que executa o cross-scripting é chamado de *parte de acesso* (normalmente o SWF que faz o acesso) e o outro lado é chamado de *a parte que está sendo acessada* (normalmente o SWF que está sendo acessado). Quando o siteA.swf executa script no siteB.swf, o siteA.swf é a parte que acessa e o siteB.swf é a parte que está sendo acessada, conforme mostrado na ilustração a seguir:



Permissões entre domínios estabelecidas com o método`Security.allowDomain()` são assimétricas. No exemplo anterior, o siteA.swf pode executar script no siteB.swf, mas o siteB.swf não pode executar script no siteA.swf, pois o siteA.swf não chamou o método `Security.allowDomain()` para conceder permissão aos arquivos SWF no siteB.com para executar script nele. É possível configurar permissões simétricas fazendo com que os dois arquivos SWF chamem o método `Security.allowDomain()`.

Além de proteger arquivos SWF contra a execução de script entre domínios originadas por outros arquivos SWF, o Flash Player protege arquivos SWF conta a execução de script entre domínios originadas por arquivos HTML. A execução de script de HTML para SWF pode ocorrer com retornos de chamada estabelecidos por meio do método `ExternalInterface.addCallback()`. Quando a execução de script de HTML para SWF cruza domínios, o arquivo SWF que está sendo acessado deve chamar o método `Security.allowDomain()` exatamente como quando a parte que faz o acesso é um arquivo SWF ou haverá falha na operação. Para obter mais informações, consulte "Controles de autor (desenvolvedor)" na página 1093.

Além disso, o Flash Player fornece controles de segurança para execução de script de SWF para HTML. Para obter mais informações, consulte "Controle do acesso à URL de saída" na página 1112.

segurança de Palco

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Algumas propriedades e métodos do objeto Stage estão disponíveis para qualquer entidade gráfica ou clipe de filme na lista de exibição.

No entanto o objeto Stage é considerado como tendo um proprietário: o primeiro arquivo SWF carregado. Por padrão, as seguintes propriedades e métodos do objeto Stage estão disponíveis apenas para arquivos SWF na mesma caixa de proteção de segurança que o proprietário do Palco:

Propriedades	Métodos
`alinhar`	`addChild()`
displayState	`addChildAt()`
frameRate	`addEventListener()`
`height`	`dispatchEvent()`
`mouseChildren`	`hasEventListener()`
`numChildren`	`setChildIndex()`
`quality`	`willTrigger()`
`scaleMode`	
`showDefaultContextMenu`	
`stageFocusRect`	
`stageHeight`	
`stageWidth`	
`tabChildren`	
`textSnapshot`	
`width`	

Para que um arquivo SWF em uma caixa de proteção diferente daquela do proprietário do Palco acesse essas propriedades e métodos, o arquivo SWF do proprietário do Palco deve chamar o método `Security.allowDomain()` para permitir o domínio da caixa de proteção externa. Para obter mais informações, consulte "Controles de autor (desenvolvedor)" na página 1093.

A propriedade `frameRate` é um caso especial, qualquer arquivo SWF pode ler a propriedade `frameRate`. No entanto, apenas aqueles que estão na caixa de proteção de segurança do proprietário do Palco (ou aqueles que receberam permissão por uma chamada para o método `Security.allowDomain()`) podem alterar a propriedade.

Também há restrições nos métodos `removeChildAt()` e `swapChildrenAt()` do objeto Stage, mas essas são diferentes das outras restrições. E vez de ser necessário estar no mesmo domínio que o proprietário do Palco para chamar esses métodos, o código deve estar no mesmo domínio que o proprietário dos objetos filho afetados ou os objetos filho podem chamar o método `Security.allowDomain()`.

Como percorrer a lista de exibição

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A capacidade de um arquivo SWF acessar objetos de exibição carregados de outras caixas de proteção é restrita. Para que um arquivo SWF acesse um objeto de exibição criado por outro arquivo SWF em uma caixa de proteção diferente, o arquivo SWF que está sendo acessado deve chamar o método `Security.allowDomain()` para permitir acesso pelo domínio do arquivo SWF que está acessando. Para obter mais informações, consulte "Controles de autor (desenvolvedor)" na página 1093.

Para acessar um objeto Bitmap que foi carregado por um objeto Loader, um arquivo de política de URL deve existir no servidor de origem do arquivo de imagem, e esse arquivo de política deve conceder permissão ao domínio do arquivo SWF que está tentando acessar o objeto Bitmap (consulte "Controles de site (arquivos de política)" na página 1089).

O objeto LoaderInfo que corresponde a um arquivo carregado (e ao objeto Loader) inclui as três seguintes propriedades que definem o relacionamento entre o objeto carregado e o objeto Loader: `childAllowsParent`, `parentAllowsChild` e `sameDomain`.

Segurança de eventos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Eventos relacionados à lista de exibição têm limitações de acesso de segurança com base na caixa de proteção do objeto de exibição está despachando o evento. Um evento na lista de exibição tem fases de bolhas e de captura (descritas em "Manipulação de eventos" na página 126). Durante as fases de bolhas e de captura, um evento migra do objeto de exibição de origem por meio de objetos de exibição pai na lista de exibição. Se um objeto pai estiver em uma caixa de proteção de segurança diferente do objeto de exibição de origem, a fase de captura e de bubble parará abaixo daquele objeto pai, a menos que haja confiança mútua entre o proprietário do objeto pai e o proprietário do objeto de origem. Essa confiança mútua pode ser obtida pelo seguinte:

1. O arquivo SWF que possui o objeto pai deve chamar o método `Security.allowDomain()` para confiar no domínio do arquivo SWF que possui o objeto de origem.
2. O arquivo SWF que possui o objeto de origem deve chamar o método `Security.allowDomain()` para confiar no domínio do arquivo SWF que possui o objeto pai.

O objeto LoaderInfo que corresponde a um arquivo carregado (e ao objeto Loader) inclui as duas seguintes propriedades que definem o relacionamento entre o objeto carregado e o objeto Loader: `childAllowsParent` e `parentAllowsChild`.

Para eventos despachados de objetos que não são objetos de exibição, não há verificações de segurança ou implicações relacionadas à segurança.

Acesso à mídia carregada como dados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para acessar dados de carregamento use os métodos `BitmapData.draw()`, `BitmapData.drawWithQuality()` e `SoundMixer.computeSpectrum()`. Por padrão, não é possível obter dados de pixel ou dados de áudio a partir de objetos gráficos ou de áudio renderizados ou reproduzidos por mídia carregada em uma área de segurança diferente. No entanto, é possível utilizar os métodos a seguir para conceder permissões para acessar estes dados nos limites de áreas de segurança:

- No conteúdo renderizado ou que está reproduzindo os dados, crie uma chamada para o método `Security.allowDomain()` para conceder acesso de dados ao conteúdo em outros domínios.
- Para uma imagem, som ou vídeo carregado, adicione um arquivo de política de URL no servidor do arquivo carregado. Esse arquivo de política deve conceder acesso ao domínio do arquivo SWF que está tentando chamar os métodos `BitmapData.draw()`, `BitmapData.drawWithQuality()` ou `SoundMixer.computeSpectrum()` para extrair dados do arquivo. O método `drawWithQuality` está disponível no Flash Player 11.3 e superior; AIR 3.3 e superior.

As seções a seguir fornecem detalhes sobre o acesso a dados de bitmap, som e vídeo.

Acesso a dados de bitmap

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Os métodos `draw()` e `drawWithQuality()` (Flash Player 11.3; AIR 3.3) de um objeto BitmapData permite que você desenhe os pixels exibidos no momento de qualquer objeto de exibição no BitmapData. Isso pode incluir os pixels de um objeto MovieClip, de um objeto Bitmap ou de qualquer objeto de exibição. As condições a seguir devem ser atendidas para que esses métodos desenhem pixels no objeto BitmapData:

- No caso de um objeto de origem diferente de um bitmap carregado, o objeto de origem e (no caso de um objeto Sprite ou MovieClip) todos os seus objetos filho devem vir do mesmo domínio que o objeto que está chamando o método draw(), ou devem estar em um arquivo SWF que possa ser acessado pelo chamador fazendo com que o método `Security.allowDomain()` seja chamado.
- No caso de um objeto de origem de bitmap Loaded, o objeto de origem deve vir do mesmo domínio que o objeto que está chamando o método draw(), ou seu servidor de origem deve incluir um arquivo de política de URL que conceda permissão ao domínio que está fazendo a chamada.

Se essas condições não forem atendidas, uma exceção SecurityError será lançada.

Ao carregar a imagem usando o método `load()` da classe Loader, você pode especificar um parâmetro `context` que é um objeto SoundLoaderContext. Se você definir a propriedade `checkPolicyFile` do objeto LoaderContext como `true`, o Flash Player verificará se há um arquivo de política de URL no servidor do qual a imagem é carregada. Se houver um arquivo de política e o arquivo permitir que o domínio do arquivo SWF que está sendo carregado, o arquivo terá permissão para acessar os dados no objeto Bitmap. Caso contrário, o acesso será negado.

Também é possível especificar uma propriedade `checkPolicyFile` em uma imagem carregada por meio de uma tag `<img>` em um campo de texto. Para obter detalhes, consulte "Carregamento de arquivos SWF e de imagens usando a tag <img> em um campo de texto" na página 1100

Acesso a dados de som

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

As seguintes APIs do ActionScript 3.0 relacionadas a som têm restrições de segurança:

- O método `SoundMixer.computeSpectrum()` — sempre permitido para arquivos SWF que estão na mesma área de proteção de segurança que o arquivo de som. Existem verificações de segurança para o código executado em outras áreas de segurança.
- O método `SoundMixer.computeSpectrum()` — sempre permitido para código sendo executado na mesma área de segurança que o arquivo de som. Para arquivos em outras caixas de proteção, há verificações de segurança.
- A propriedade `id3` da classe Sound — sempre permitida para arquivos SWF que estão na mesma caixa de proteção de segurança que o arquivo de som. Existem verificações de segurança para o código executado em outras áreas de segurança.

Todo evento tem dois tipos de caixas de proteção associadas a ele, uma caixa de proteção de conteúdo e uma caixa de proteção do proprietário:

- O domínio de origem do som determina a caixa de proteção de conteúdo e isso determina se os dados podem ser extraídos do som por meio da propriedade `id3` do som e do método `SoundMixer.computeSpectrum()`.
- O objeto que iniciou a reprodução do som determina a caixa de proteção do proprietário e isso determina se o som pode ser parado usando o método `SoundMixer.stopAll()`.

Ao carregar o som usando o método `load()` da classe Sound, você pode especificar um parâmetro `context` que é um objeto SoundLoaderContext. Se você definir a propriedade `checkPolicyFile` do objeto SoundLoaderContext como `true`, o Flash Player verificará se há um arquivo de política de URL no servidor do qual o som é carregado. Se houver um arquivo de política e o arquivo permitir o domínio do código que está sendo carregado, o código receberá permissão para acessar a propriedade `id` do objeto Sound. Caso contrário, não terá essa permissão. Além disso, a configuração da propriedade `checkPolicyFile` pode ativar o método `SoundMixer.computeSpectrum()` para sons carregados.

É possível usar o método `SoundMixer.areSoundsInaccessible()` para descobrir se uma chamada para o método `SoundMixer.stopAll()` não parará todos os sons porque a caixa de proteção de um ou mais proprietários de som está inacessível para o chamador.

A chamada do método `SoundMixer.stopAll()` para esses sons cuja caixa de proteção de proprietário é a mesma que a do chamador de `stopAll()`. Ela também para aqueles sons cuja reprodução foi iniciada por arquivos SWF que chamaram o método `Security.allowDomain()` para permitir acesso pelo domínio do arquivo SWF que está chamando o método `stopAll()`. Qualquer outro som não é parado e a presença desses sons pode ser revelada chamando o método `SoundMixer.areSoundsInaccessible()`.

A chamada do método `computeSpectrum()` requer que todo som que esteja sendo reproduzido seja da mesma caixa de proteção que o objeto que está chamando o método ou de uma origem que recebeu permissão para a caixa de proteção do chamador. Caso contrário, uma exceção SecurityError é lançada. Para sons que foram carregados de sons incorporados em uma biblioteca em um arquivo SWF, a permissão será concedida com uma chamada para o método `Security.allowDomain()` no arquivo SWF carregado. Para sons carregados de origens que não sejam arquivos SWF (originários de arquivos mp3 carregados ou de arquivos de vídeo), um arquivo de política de URL no servidor de origem concede acesso a dados contidos na mídia carregada.

Para obter mais informações, consulte“Controles de autor (desenvolvedor)” na página 1093 e “Controles de site (arquivos de política)” na página 1089.

Para acessar dados de som a partir de fluxos RTMP, é necessário conceder acesso no servidor. Utilize a propriedade ActionScript de lado do servidor `Client.audioSampleAccess` para permitir acesso a diretórios específicos no Flash Media Server. Para obter mais informações, consulte a Referência de linguagem do ActionScript de lado do servidor.

Acesso a dados de vídeo

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Você pode usar o método `BitmapData.draw()` ou `BitmapData.drawWithQuality()` para capturar os dados de pixel o quadro atual de um vídeo. (O método `drawWithQuality` está disponível no Flash Player 11.3 e superior; AIR 3.3 e superior.)

Há dois tipos diferentes de vídeo:

- Vídeo transmitido por meio de RTMP a partir do Flash Media Server
- Vídeo progressivo, carregado a partir de um arquivo FLV ou F4V

Para usar métodos de desenho e extrair gráficos de tempo de execução de fluxos RTMP, você deve permitir o acesso no servidor. Utilize a propriedade ActionScript de lado do servidor Client.videoSampleAccess para permitir acesso a diretórios específicos no Flash Media Server. Para obter mais informações, consulte a Referência de linguagem do ActionScript de lado do servidor.

Ao chamar um método de desenho com vídeo progressivo como parâmetro de `origem`, o chamador do método deve ser da mesma segurança que o arquivo FLV ou o servidor do arquivo FLV deve ter um arquivo de política que concede permissão ao domínio do arquivo chamador SWF. É possível solicitar que o arquivo de política seja baixado configurando a propriedade `checkPolicyFile` do objeto NetStream como `true`.

Carregamento de dados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O conteúdo do Flash Player e do AIR podem trocar dados com servidores. O carregamento de dados é um tipo de operação diferente do carregamento de mídia porque as informações carregadas aparecem como objetos em vez de serem exibidas como mídia. Geralmente, o conteúdo pode carregar dados do mesmo domínio de origem do conteúdo. No entanto, o conteúdo geralmente exige arquivos de política para carregar dados de outros domínios (consulte "Controles de site (arquivos de política)" na página 1089).

Nota: *O conteúdo executado na área de segurança do AIR nunca é servidor a partir de um domínio remoto (a não ser que o desenvolvedor importe intencionalmente o conteúdo remoto para a área de segurança), de forma que ele não participe da proteção contra ataques fornecida pelos arquivos de política. O conteúdo do AIR na área de segurança do aplicativo não está restrito a carregar dados por meio de arquivos de política. No entanto, o conteúdo AIR em outras áreas de segurança está sujeito às restrições descritas.*

Uso de URLLoader e URLStream

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

É possível carregar dados, como um arquivo XML ou um arquivo de texto. Os métodos `load()` das classes URLLoader e URLStream são governados pelas permissões do arquivo de política de URL.

Se você usar o método `load()` para carregar conteúdo de um domínio diferente daquele do código que está chamando o método, o tempo de execução verificará se existe um arquivo de política de URL no servidor dos ativos carregados. Se houver um arquivo de política e ele conceder acesso ao domínio do conteúdo que está sendo carregado, você poderá carregar os dados.

Conexão a soquetes

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Por padrão, o tempo de execução procura um arquivo de política de soquete servido na porta 843. Assim como os arquivos de política de URL, esse arquivo é chamado de *arquivo de política mestre*.

Quando arquivos de política foram introduzidos no Flash Player 6 pela primeira vez, não havia suporte para arquivos de política de soquete. Conexões a servidores de soquete eram autorizadas por um arquivo de política no local padrão em um servidor HTTP na porta 80 do mesmo host que o servidor de soquete. O Flash Player 9 ainda oferece suporte a essa capacidade, mas o Flash Player 10 não oferece. No Flash Player 10, apenas arquivos de política de soquete podem autorizar conexões de soquete.

Como os arquivos de política de URL, os arquivos de política de soquete oferecem suporte a uma instrução de metapolítica que especifica quais portas podem servir arquivos de política. No entanto, em vez de “master-only”, a metapolítica padrão para arquivos de política de soquete é “all”. Isto é, a menos que o arquivo de política mestre especifique uma configuração mais restritiva, o Flash Player assumirá que qualquer soquete no host pode servir um arquivo de política de soquete.

O acesso a conexões de soquete XML e a soquete está desativado por padrão, mesmo que o soquete que você está conectando esteja no mesmo domínio que o arquivo SWF. É possível permitir acesso em nível de soquete servindo um arquivo de política de segurança de qualquer um dos seguintes locais:

- Porta 843 (o local do arquivo de política mestre)
- A mesma porta que a conexão de soquete principal
- Uma porta diferente da conexão de soquete principal

Por padrão, o Flash Player procura um arquivo de política de soquete na porta 843 e na mesma porta que a conexão de soquete principal. Se desejar servir um arquivo de política de soquete em uma porta diferente, o arquivo SWF deverá chamar `Security.loadPolicyFile()`.

Um arquivo de política de soquete tem a mesma sintaxe que um arquivo de política de URL, exceto que ele também deve especificar as portas às quais ele concede acesso. Quando um arquivo de política de soquete é servido em uma porta com número inferior a 1024, ele pode conceder acesso a qualquer porta. Quando um arquivo de política é proveniente da porta 1024 ou superior ele só pode conceder acesso a portas 1024 e superiores. As portas permitidas são especificadas em um atributo `to-ports` na tag `<allow-access-from>`. Os valores de números de portas únicos, intervalos de portas e curingas são aceitos.

Este é um exemplo de arquivo de política de soquete:

```
<?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>
```

Para recuperar um arquivo de política de soquete da porta 843 ou da mesma porta como uma conexão de soquete principal, chame o método `Socket.connect()` ou `XMLSocket.connect()`. O Flash Player primeiro verifica se há um arquivo de política mestre na porta 843. Se um arquivo de política mestre for localizado, ele verificará se o arquivo contém uma instrução de metapolítica que proíbe arquivos de política de soquete na porta de destino. Se o acesso não for proibido, o Flash Player primeiro procurará a instrução `allow-access-from` apropriada no arquivo de política mestre. Se uma instrução não for localizada, ele procurará um arquivo de política de soquete na mesma porta que a conexão de soquete principal.

Para recuperar um arquivo de política de soquete de um local diferente, primeiro chame o método `Security.loadPolicyFile()` com a sintaxe especial `"xmlsocket"`, como no seguinte:

```
Security.loadPolicyFile("xmlsocket://server.com:2525");
```

Chame o método `Security.loadPolicyFile()` antes de chamar o método `Socket.connect()` ou `XMLSocket.connect()`. O Flash Player então aguarda até que tenha atendido sua solicitação de arquivo de política antes de decidir se permite sua conexão principal. No entanto, se o arquivo de política mestre especificar que o local de destino não pode servir arquivos de política, a chamada para `loadPolicyFile()` não terá nenhum efeito, mesmo que haja um arquivo de política naquele local.

Se estiver implementando um servidor de soquetes e precisar fornecer um arquivo de política de soquete, decida se o arquivo de política deve ser fornecido usando a mesma porta que aceita conexões principais ou usando uma porta diferente. Em qualquer dos casos, o servidor deve aguardar a primeira transmissão de seu cliente antes de enviar uma resposta.

Quando o Flash Player solicita um arquivo de política, ele sempre transmite a string a seguir assim que a conexão é estabelecida:

```
<policy-file-request/>
```

Quando o servidor recebe essa string, ele pode transmitir o arquivo de política. A solicitação do Flash Player sempre é terminada por um byte nulo e a resposta do servidor também deve ser terminada por um byte nulo.

Não espere reutilizar a mesma conexão para uma solicitação de arquivo de política e para uma conexão principal. Feche a conexão depois de transmitir o arquivo de política. Se isso não for feito, o Flash Player fechará a conexão do arquivo de política antes de reconectar-se para configurar a conexão principal.

Protegendo dados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para proteger o vazamento de dados e alterações durante o transporte na Internet, é possível utilizar Transport Layer Security (TLS) ou Socket Layer Security (SSL) no servidor de origem dos dados. É possível estabelecer uma conexão com o servidor utilizando o protocolo HTTPS.

Em aplicativos criados no AIR 2 ou superior, você também pode proteger as comunicações do soquete TCP. A classe SecureSocket permite que você inicie uma conexão com um servidor de soquete que utiliza TLS versão 1 ou SSL versão 4.

Envio de dados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O envio de dados ocorre quando o código envia dados para um servidor ou recurso. O envio de dados é sempre permitido para o conteúdo de uma domínio de rede. Um arquivo SWF local pode enviar dados a endereços de rede somente se estiver na área de proteção local confiável, local com rede ou em uma área de segurança AIR. Para obter mais informações, consulte "Caixas de proteção locais" na página 1081.

É possível usar a função `flash.net.sendToURL()` para enviar dados a uma URL. Outros métodos também enviam solicitações a URLs. Esses incluem métodos de carregamento, como `Loader.load()` e `Sound.load()`, métodos de carregamento de dados, como `URLLoader.load()` e `URLStream.load()`.

Upload e download de arquivos

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O método `FileReference.upload()` inicia o upload de um arquivo selecionado por um usuário a um servidor remoto. É necessário chamar o método `FileReference.browse()` ou `FileReferenceList.browse()` antes de chamar o método `FileReference.upload()`.

O código que inicia o método `FileReference.browse()` ou `FileReferenceList.browse()` somente pode ser chamado em resposta a um evento de mouse ou de teclado. Se ele for chamado em outras situações, o Flash Player 10 e versões posteriores irão gerar uma exceção. No entanto, um evento iniciado por usuário não é obrigatório para chamar estes métodos a partir da área de segurança do aplicativo do AIR.

A chamada do método `FileReference.download()` abre a caixa de diálogo na qual o usuário pode baixar um arquivo de um servidor remoto.

Nota: *Se o servidor exigir autenticação de usuário, apenas os aplicativos Flash em execução em um navegador (ou seja, que utilizam plug-in para navegador ou controle ActiveX), poderão fornecer uma caixa de diálogo para solicitar que o usuário insira um nome e uma senha para autenticação, e somente para downloads. O Flash Player não permite uploads para um servidor que exige autenticação de usuário.*

Uploads e downloads não serão permitidos se o arquivo SWF que está chamando estiver na caixa de proteção local com sistema de arquivos.

Por padrão, um arquivo SWF não pode iniciar um upload ou um download de um servidor que não seja seu próprio servidor. Um arquivo SWF pode carregar ou baixar de um servidor diferente se esse servidor fornecer um arquivo de política que conceda permissão ao domínio do arquivo SWF que está chamando.

Carregamento de conteúdo incorporado de arquivos SWF importados em um domínio de segurança

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Ao carregar um arquivo SWF, você pode definir o parâmetro `context` do método `load()` do objeto Loader usado para carregar o arquivo. Esse parâmetro usa um objeto LoaderContext. Ao definir a propriedade `securityDomain` do objeto LoaderContext como `Security.currentDomain`, o Flash Player verificará se há um arquivo de política de URL no servidor do arquivo SWF carregado. Se houver um arquivo de política e ele conceder acesso ao domínio do arquivo SWF que está sendo carregado, você poderá carregar o arquivo SWF como mídia importada. Dessa maneira, o arquivo carregado pode obter acesso a objetos na biblioteca do arquivo SWF.

Uma maneira alternativa de um arquivo SWF acessar classes no arquivo SWF carregado de uma caixa de proteção de segurança diferente é fazer com que o arquivo SWF carregado chame o método `Security.allowDomain()` para conceder acesso ao domínio do arquivo SWF que está chamando. É possível adicionar a chamada ao método `Security.allowDomain()` ao método construtor da classe principal do arquivo SWF carregado e fazer com que o arquivo SWF que está sendo carregado adicione um ouvinte de eventos para responder ao evento `init` despachado pela propriedade `contentLoaderInfo` do objeto Loader. Quando esse evento for despachado, o arquivo SWF carregado terá chamado o método `Security.allowDomain()` no método construtor e as classes no arquivo SWF carregado estarão disponíveis para o arquivo SWF que está sendo carregado. O arquivo SWF sendo carregado pode recuperar as classes do arquivo SWF carregado chamando `Loader.contentLoaderInfo.applicationDomain.getDefinition()` ou Loader.contentLoaderInfo.applicationDomain.getQualifiedDefinitionNames()(Flash Player 11.3 e superior; AIR 3.3 e superior).

Trabalho com conteúdo legado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

No Flash Player 6, o domínio usado para determinadas configurações do Flash Player se baseia na parte posterior do domínio do arquivo SWF. Essas configurações incluem configurações para permissões de câmera e microfone, quotas de armazenamento e armazenamento de objetos compartilhados persistentes.

Se o domínio de um arquivo SWF incluir mais de dois segmentos, como www.example.com, o primeiro segmento do domínio (www) será removido e a parte restante do domínio será usada. Assim, no Flash Player 6, www.example.com e store.example.com usam example.com como domínio dessas configurações. De modo semelhante, www.example.co.uk e store.example.co.uk usam example.co.uk como o domínio dessas configurações. Isso pode resultar em problemas nos quais arquivos SWF de domínios não relacionados, como example1.co.uk e example2.co.uk, têm acesso aos mesmos objetos compartilhados.

No Flash Player 7 e posterior, as configurações do player são escolhidas por padrão de acordo com o domínio exato de um arquivo SWF. Por exemplo, um arquivo SWF do www.example.com usa as configurações do player para www.example.com. Um arquivo SWF do store.example.com usa as configurações separadas do player para store.example.com.

Em um arquivo SWF gravado com o ActionScript 3.0, quando `Security.exactSettings` é definida como `true` (o padrão), o Flash Player usa domínios exatos para configurações do player. Quando está definido como `false`, o Flash Player usa as configurações de domínio utilizadas no Flash Player 6. Se você alterar `exactSettings` de seu valor padrão, faça isso antes que ocorram quaisquer eventos que exigem que o Flash Player escolha as configurações do player — por exemplo, usando uma câmera ou microfone, ou recuperando um objeto compartilhado consistente.

Se você publicou um arquivo SWF da versão 6 e criou objetos compartilhados persistentes a partir dele, para recuperar esses objetos compartilhados persistentes de um SWF que usa o ActionScript 3.0, você deve definir `Security.exactSettings` como `false` antes de chamar `SharedObject.getLocal()`.

Configuração de permissões de LocalConnection

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

A classe LocalConnection permite enviar mensagens entre um Flash Player ou um aplicativo do AIR. Objetos LocalConnection podem comunicar-se somente entre o conteúdo do Flash Player ou AIR executado no mesmo computador, mas eles podem executar aplicativos diferentes—por exemplo, um arquivo SWF executado em um navegador, um arquivo SWF executado em um projetor e um aplicativo do AIR podem se comunicar utilizando a classe LocalConnection.

Para cada comunicação LocalConnection, existe um remetente e um ouvinte. Por padrão, o Flash Player permite comunicação LocalConnection entre o código executado no mesmo domínio. Para código executado em diferentes áreas de proteção, o ouvinte deve conceder a permissão de remetente, usando o método `LocalConnection.allowDomain()`. A string passada como um argumento para o método `LocalConnection.allowDomain()` pode conter qualquer um dos seguintes: nomes exatos do domínio, endereços IP e `*` asterisco.

A forma do método `allowDomain()` foi alterada desde o ActionScript 1.0 e 2.0. Nessas versões anteriores, `allowDomain()` era um método de retorno de chamada que você implementava. No ActionScript 3.0, `allowDomain()` é um método embutido da classe LocalConnection que você chama. Com essa alteração, `allowDomain()` funciona da mesma maneira que `Security.allowDomain()`.

Um arquivo SWF usa a propriedade `domain` da classe LocalConnection para determinar seu domínio.

Controle do acesso à URL de saída

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Script e acesso à URL de saída (por meio do uso de URLs HTTP, mailto: etc.) são executados com o uso das seguintes APIs do

- A função `flash.system.fscommand()`
- O método `ExternalInterface.call()`
- A função `flash.net.navigateToURL()`

Para o conteúdo carregado a partir do sistema de arquivos local, as chamadas para estes métodos são bem sucedidas se o código e a página web (se houver uma) estiverem nas áreas de segurança do aplicativo do AIR. Haverá falha em chamadas para esses métodos se o conteúdo estiver na caixa de proteção local com rede ou local com sistema de arquivos.

Para conteúdo não carregado localmente, todas essas APIs podem se comunicar com a página da Web na qual estão incorporados, dependendo do valor do parâmetro AllowScriptAccess descrito a seguir. A função `flash.net.navigateToURL()` tem a capacidade adicional de comunicação com qualquer janela ou quadro de navegador aberto, não apenas com a página na qual o arquivo SWF está incorporado. Para obter mais informações sobre essa funcionalidade, consulte "Uso da função navigateToURL()" na página 1113.

O parâmetro `AllowScriptAccess` no código HTML que carrega um arquivo SWF controla a capacidade de executar acesso à URL a partir do arquivo SWF. Defina esse parâmetro dentro da tag PARAM ou EMBED. Se nenhum valor estiver definido para `AllowScriptAccess`, o arquivo SWF e a página HTML poderão se comunicar apenas se os dois forem do mesmo domínio.

O valor do parâmetro `AllowScriptAccess` pode ser um destes três valores possíveis: `"always"`, `"sameDomain"` ou `"never"`.

- Quando `AllowScriptAccess` é `"always"`, o arquivo SWF pode se comunicar com a página HTML na qual ele está incorporado, mesmo quando o arquivo é de um domínio diferente da página HTML.
- Quando `AllowScriptAccess` é `"sameDomain"`, o arquivo SWF pode se comunicar com a página HTML na qual ele está incorporado apenas quando o arquivo SWF está no mesmo domínio que a página HTML. Esse valor é o padrão de `AllowScriptAccess`. Use essa configuração, ou não defina um valor para `AllowScriptAccess`, para evitar que um arquivo SWF hospedado em um domínio acesse um script em uma página HTML proveniente de outro domínio.
- Quando `AllowScriptAccess` está definido como `"never"`, um arquivo SWF não pode se comunicar com nenhuma página HTML. Usar este valor foi algo desaprovado deste o lançamento do Adobe Flash CS4 Professional. Ele não é recomendado e não deveria ser necessário se você não disponibiliza arquivos SWF não confiáveis de seu próprio domínio. Se for necessário servir arquivos SWF não confiáveis, a Adobe recomenda que seja criado um subdomínio diferente e que todo o conteúdo não confiável seja colocado nele.

Este é um exemplo de configuração da tag `AllowScriptAccess` em uma página HTML para permitir acesso à URL de saída para um domínio diferente:

```
<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>
```

Uso da função navigateToURL()

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Além da configuração de segurança especificada pelo parâmetro `allowScriptAccess` discutido acima, a função `navigateToURL()` tem um segundo parâmetro - `target`. O parâmetro `target` pode ser usado para especificar o nome de uma janela ou quadro HTML para o qual enviar a solicitação de URL. Restrições de segurança adicionais se aplicam a essas solicitações, e as restrições variarão se `navigateToURL()` estiver sendo usado como uma instrução de script ou de não script.

Para instruções de script, como `navigateToURL("javascript: alert('Hello from Flash Player.')")`, as seguintes regras são aplicadas.

- Se o arquivo SWF for um arquivo confiável localmente, a solicitação terá êxito.

- Se o destino for a página HTML na qual o arquivo SWF está incorporado, as regras de `allowScriptAccess` descritas acima serão aplicadas.
- Se o destino contiver conteúdo carregado do mesmo domínio que o arquivo SWF, a solicitação terá êxito.
- Se o destino contiver conteúdo carregado de um domínio diferente do arquivo SWF, e nenhuma das duas condições anteriores for atendida, haverá falha na solicitação.

Para instruções de não script (como HTTP, HTTPS e `mailto:`, haverá falha na solicitação se todas as condições a seguir se aplicarem:

- O destino é uma das palavras-chave `"_top"` ou `"_parent"` e
- o arquivo SWF é uma página da Web hospedada em um domínio diferente e
- o arquivo SWF está incorporado com um valor de `allowScriptAccess` diferente de `"always"`.

Para obter mais informações

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Para obter mais informações sobre acesso de URL externas, consulte as entradas a seguir em *Referência do ActionScript® 3.0 para Adobe® Flash® Platform:*

- A função flash.system.fscommand()
- O método da classe ExternalInterfacecall()
- A função `flash.net.navigateToURL()`

Objetos compartilhados

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

O Flash Player fornece a capacidade de usar *objetos compartilhados* que são objetos ActionScript que persistem fora de um arquivo SWF, seja localmente no sistema de arquivos de um usuário ou remotamente em um servidor RTP. Objetos compartilhados, como outras mídias no Flash Player, são particionados em caixas de proteção de segurança. No entanto o modelo de caixa de proteção para objetos compartilhados é um pouco diferente porque objetos compartilhados não são recursos que podem ser acessados além dos limites do domínio. Ao contrário, os objetos compartilhados sempre são recuperados de um armazenamento de objetos compartilhados que é específico ao domínio de cada arquivo SWF que chama métodos da classe SharedObject. Normalmente, um objeto compartilhado é mais particular do que o domínio de um arquivo SWF: por padrão, cada arquivo SWF usa um armazenamento de objetos compartilhados específico a toda a sua URL de origem. Para obter mais informações sobre objetos compartilhados, consulte “Objetos compartilhados” na página 724.

Um arquivo SWF pode usar o parâmetro `localPath` dos métodos `SharedObject.getLocal()` e `SharedObject.getRemote()` para usar um armazenamento de objetos compartilhados com apenas uma parte de sua URL. Dessa maneira, o arquivo SWF pode permitir o compartilhamento com outros arquivos SWF de outras URLs. Mesmo que você passe `'/'` como o parâmetro `localPath`, ele ainda especifica um objeto compartilhado específico a seu próprio domínio.

Os usuários podem restringir o acesso a objetos compartilhados usando a caixa de diálogo Configurações do Flash Player ou o Gerenciador de configurações. Por padrão, os objetos compartilhados podem ser criados até um máximo de 100 KB de dados por domínio. Usuários administrativos e usuários também podem aplicar restrições quanto à capacidade de gravar no sistema de arquivos. Para obter mais informações, consulte "Controles de administrador" na página 1086 e "Controles de usuário" na página 1088.

É possível especificar que um objeto compartilhado está protegido especificando `true` para o parâmetro `secure` do método `SharedObject.getLocal()` ou do método `SharedObject.getRemote()`. Observe o seguinte sobre o parâmetro `secure`:

- Se esse parâmetro for definido como `true`, o Flash Player criará um novo objeto compartilhado ou obterá uma referência a um objeto compartilhado protegido existente. Esse objeto compartilhado protegido apenas poderá ser lido ou gravado por arquivos SWF entregues via HTTPS que chamarem `SharedObject.getLocal()` com o parâmetro `protegido` definido como `true`.
- Se esse parâmetro for definido como `false`, o Flash Player criará um novo objeto compartilhado ou obterá uma referência a um objeto compartilhado existente que pode ser lido ou gravado por arquivos SWF entregues via conexões não HTTPS.

Se o arquivo SWF que faz a chamada não for de uma URL HTTPS, especificar `true` para o parâmetro `secure` do método `SharedObject.getLocal()` ou do método `SharedObject.getRemote()` resultará em uma exceção SecurityError.

A opção de armazenamento de um objeto compartilhado tem como base a URL de origem do arquivo SWF. Isso é verdadeiro mesmo nas duas situações em que um arquivo SWF não é originário de uma URL simples: carregamento de importação e carregamento dinâmico. O carregamento de importação se refere à situação em que você carrega um arquivo SWF com a propriedade `LoaderContext.securityDomain` definida como `SecurityDomain.currentDomain`. Nessa situação, o arquivo SWF carregado terá uma pseudoURL que começa com o domínio dou arquivo SWF carregado e especifica sua URL de origem real. O carregamento dinâmico refere-se ao carregamento de um arquivo SWF usando o método `Loader.loadBytes()`. Nesta situação, o arquivo SWF carregado terá um pseudoURL que começa com o URL completo de seu arquivo SWF de carregamento, seguido por um ID de inteiro. Nos casos de carregamento dinâmico e e de carregamento de importação, pode-se examinar o pseudoURL do arquivo SWF usando a propriedade `LoaderInfo.url`. A pseudoURL é tratada exatamente como uma URL real para a escolha de um armazenamento de objetos compartilhados. É possível especificar o parâmetro `localPath` de um objeto compartilhado que use parte ou toda a pseudoURL.

Usuários e administradores podem optar por desativar o uso de *objetos compartilhados de terceiros*. Esse é o uso de objetos compartilhados por qualquer arquivo SWF que esteja em execução em um navegador da Web, quando a URL de origem desse arquivo SWF for de um domínio diferente da URL mostrada na barra de endereço do navegador. Usuários e administradores podem optar por desativar o uso de objetos compartilhados de terceiros por motivos de privacidade, com o objetivo de impedir rastreamento entre domínios. Para evitar essa restrição, convém garantir que qualquer arquivo SWF que usa objetos compartilhados seja carregado apenas dentro de estruturas da página HTML que garantem que o arquivo SWF se origina do mesmo domínio mostrado na barra de endereço do navegador. Quando você tenta usar objetos compartilhados de um arquivo SWF de terceiros, e o objeto compartilhado de terceiros está desativado, os métodos `SharedObject.getLocal()` e `SharedObject.getRemote()` retornam `null`. Para obter mais informações, consulte www.adobe.com/products/flashplayer/articles/thirdpartylso.

Acesso a câmera, microfone, área de transferência, mouse e teclado

Flash Player 9 e posterior, Adobe AIR 1.0 e posterior

Quando um arquivo SWF tenta acessar a câmera ou o microfone de um usuário usando os métodos `Camera.get()` ou `Microphone.get()`, o Flash Player exibe a caixa de diálogo Privacidade na qual o usuário pode permitir ou negar acesso à sua câmera e microfone. O usuário e o usuário administrativo também podem desativar o acesso à câmera em uma base por site ou global, por meio de controles no arquivo mms.cfg, da UI de Configurações e do Gerenciador de configurações (consulte "Controles de administrador" na página 1086 e "Controles de usuário" na página 1088). Com restrições de usuário, cada método `Camera.get()` e `Microphone.get()` retorna um valor `null`. É possível usar a propriedade `Capabilities.avHardwareDisable` para determinar se a câmera e o microfone foram proibidos (`true`) ou permitidos (`false`) administrativamente.

O método `System.setClipboard()` permite que um arquivo SWF substitua o conteúdo da área de transferência por uma string de caracteres de texto simples. Isso não impõe nenhum risco de segurança. Para proteção contra o risco imposto por recorte e cópia de senhas e outros dados confidenciais nas áreas de transferência, não há nenhum método `getClipboard()` correspondente.

Um aplicativo executado no Flash Player só pode monitorar eventos de teclado e de mouse que ocorrem dentro de seu foco. O conteúdo executado no Flash Player não consegue detectar eventos de teclado ou de mouse em outro aplicativo.

Segurança do AIR

Adobe AIR 1.0 e posterior

Noções básicas de segurança do AIR

Adobe AIR 1.0 e posterior

Os aplicativos AIR são executados com as mesmas restrições de segurança dos aplicativos nativos. Em geral, aplicativos AIR, como aplicativos nativos, têm amplo acesso aos recursos do sistema operacional, como leitura e gravação de arquivos, inicialização de aplicativos, desenho na tela e comunicação com a rede. As restrições do sistema operacional aplicadas aos aplicativos nativos, tais como privilégios específicos do usuário, são aplicadas da mesma forma aos aplicativos AIR.

Embora o modelo de segurança do Adobe® AIR® seja uma evolução do modelo de segurança do Adobe® Flash® Player, o contrato de segurança é diferente daquele aplicado ao conteúdo em um navegador. Esse contrato oferece aos desenvolvedores um meio seguro de funcionalidade mais ampla para experiências enriquecedoras com uma liberdade que seria inadequada para um aplicativo baseado em navegador.

Os Aplicativos AIR são gravados usando o código de bytes compilado (conteúdo SWF) ou o script interpretado (JavaScript, HTML) para que o tempo de execução forneça gerenciamento de memória. Isso minimiza as chances do aplicativo do AIR ser afetado pelas vulnerabilidades relacionadas a gerenciamento de memória, como estouro de buffer e corrupção de memória. Essas são algumas das vulnerabilidades mais comuns que afetam os aplicativos de área de trabalho gravados em código nativo.

Instalação e atualizações

Adobe AIR 1.0 e posterior

Os aplicativos AIR são distribuídos por arquivos de instalador do AIR que usam a extensão `air` ou por instaladores nativos, que usam o formato de arquivo e a extensão da plataforma nativa. Por exemplo, o formato do instalador nativo do Windows é um arquivo EXE, enquanto o do Android é um arquivo APK.

Quando o Adobe AIR é instalado e um arquivo de instalador do AIR é aberto, o tempo de execução do AIR administra o processo de instalação. Quando um instalador nativo é usado, o sistema operacional administra o processo de instalação.

Nota: *Os desenvolvedores podem especificar um nome de versão e de aplicativo e a origem de editor ao usar o formato de arquivo do AIR, mas o próprio fluxo de trabalho inicial de instalação do aplicativo não pode ser modificado. Essa restrição é vantajosa para os usuários, pois todos os aplicativos AIR compartilham um procedimento de instalação consistente, otimizado e seguro, administrado pelo tempo de execução. Se for necessária a personalização do aplicativo, ela poderá ser feita quando o aplicativo for executado pela primeira vez.*

Local de instalação do tempo de execução

Adobe AIR 1.0 e posterior

Os aplicativos AIR que usam o formato de arquivo AIR exigem que, primeiramente, o tempo de execução esteja instalado no computador do usuário, assim como os arquivos SWF exigem que, primeiramente, o plug-in do navegador do Flash Player esteja instalado.

O tempo de execução é instalado no seguinte local em computadores desktop:

- Mac OS: `/Library/Frameworks/`
- Windows: `C:\Program Files\Common Files\Adobe AIR`
- Linux: `/opt/Adobe AIR/`

No Mac OS, para instalar uma versão atualizada de um aplicativo, o usuário deve ter privilégios adequados do sistema para instalar no diretório do aplicativo. No Windows e no Linux, um usuário precisa de privilégios administrativos.

Nota: *No iOS, o tempo de execução do AIR não é instalado separadamente. Cada aplicativo do AIR é um aplicativo autônomo.*

O tempo de execução pode ser instalado de duas maneiras: usando o recurso de instalação direta (instalando diretamente do navegador da Web) ou por meio de instalação manual. Os aplicativos AIR empacotados como instaladores nativos também podem instalar o tempo de execução AIR como parte de seu processo de instalação normal de aplicativo. (A distribuição do tempo de execução do AIR desta forma precisa de um contrato de redistribuição com a Adobe.)

Instalação direta (tempo de execução e aplicativo)

Adobe AIR 1.0 e posterior

O recurso de instalação direta fornece aos desenvolvedores uma experiência de instalação aprimorada para os usuários que ainda não têm o Adobe AIR instalado. No método de instalação direta, o desenvolvedor cria um arquivo SWF que apresenta o aplicativo de instalação. Quando o usuário clica no arquivo SWF para instalar o aplicativo, ele tenta detectar o tempo de execução. Se o tempo de execução não puder ser detectado, ele será instalado e o tempo de execução será ativado imediatamente com o processo de instalação do aplicativo do desenvolvedor.

Instalação manual

Adobe AIR 1.0 e posterior

Se desejar, o usuário pode fazer o download e a instalação manual do tempo de execução antes de abrir o arquivo AIR. Em seguida, o desenvolvedor pode distribuir o arquivo AIR por meios diversos (por exemplo, por email ou link HTML em um site da Web). Quando o arquivo AIR é aberto, o tempo de execução inicia o processo de instalação do aplicativo.

Fluxo de instalação do aplicativo

Adobe AIR 1.0 e posterior

O modelo de segurança do AIR permite que os usuários decidam se devem instalar o aplicativo do AIR . A experiência de instalação do AIR oferece diversas melhorias sobre as tecnologias de instalação de aplicativo nativo, que tornam essa decisão de confiança mais fácil para usuários:

- O tempo de execução fornece uma experiência de instalação consistente em todos os sistemas operacionais, mesmo quando o aplicativo do AIR é instalado de um link em um navegador da Web. A maioria das experiências de instalação de aplicativo nativo dependem do navegador ou de outro aplicativo para fornecer informações de segurança, se de fato elas são fornecidas.
- A experiência de instalação do aplicativo do AIR identifica a fonte do aplicativo e as informações sobre que privilégios estão disponíveis para o aplicativo (se o usuário permitir que a instalação continue).
- O tempo de execução administra o processo de instalação de um aplicativo do AIR . O aplicativo do AIR não pode manipular o processo de instalação que o tempo de execução usa.

Em geral, os usuários não devem instalar nenhum aplicativo de área de trabalho vindo de uma fonte não confiável ou que não possa ser verificada. O ônus da prova de segurança de aplicativos nativos é, da mesma forma, verdadeiro para aplicativos AIR, assim como é para outros aplicativos instaláveis.

Destino do aplicativo

Adobe AIR 1.0 e posterior

O diretório de instalação pode ser definido usando uma das duas opções a seguir:

1. O usuário personaliza o destino durante a instalação. O aplicativo é instalado onde o usuário especificar.
2. Se o usuário não alterar o destino da instalação, o aplicativo será instalado no caminho padrão, conforme determinado pelo tempo de execução:
   - Mac OS: `~/Applications/`
   - Windows XP e anterior: `C:\Program Files\`
   - Windows Vista: `~/Apps/`
   - Linux: /opt/

   Se o desenvolvedor especificar a configuração `installFolder` no arquivo descritor do aplicativo, o aplicativo será instalado em um subcaminho desse diretório.

O sistema de arquivos AIR

Adobe AIR 1.0 e posterior

O processo de instalação de aplicativos AIR copia todos os arquivos que o desenvolvedor incluiu no arquivo do instalador do AIR para o computador local do usuário. O aplicativo instalado é composto de:

- Windows: Um diretório contendo todos os arquivos incluídos no arquivo do instalador AIR. O tempo de execução também cria um arquivo exe durante a instalação do aplicativo do AIR.
- Linux: Um diretório contendo todos os arquivos incluídos no arquivo do instalador do AIR. O tempo de execução também cria um arquivo bin durante a instalação do aplicativo do AIR.
- Mac OS: Um arquivo `app` que contém todo o conteúdo do arquivo do instalador AIR. Ele pode ser inspecionado usando a opção "Mostrar conteúdo do pacote" do Localizador. O tempo de execução cria esse arquivo app como parte da instalação do aplicativo do AIR.

O aplicativo do AIR é executado por:

- Windows: Execução do arquivo .exe na pasta de instalação ou um atalho que corresponda a esse arquivo (como um atalho do menu Iniciar ou área de trabalho).
- Linux: Iniciar o arquivo .bin na pasta de instalação, selecionar o aplicativo no menu Aplicativos ou executar um alias ou atalho da área de trabalho.
- Mac OS: Execução do arquivo .app ou um alias que aponte para ele.

O sistema de arquivos do aplicativo também inclui subdiretórios relacionados à função do aplicativo. Por exemplo, as informações gravadas no depósito local criptografado são salvas em um subdiretório do diretório nomeado depois do identificador de aplicativo do aplicativo.

Armazenamento de aplicativo do AIR

Adobe AIR 1.0 e posterior

Os aplicativos AIR têm privilégios de gravação em qualquer local do disco rígido do usuário, contudo, os desenvolvedores são incentivados a usar o caminho `app-storage:/` para armazenamento local relacionado aos respectivos aplicativos. Os arquivos gravados em `app-storage:/` de um aplicativo ficam localizados em um local padrão, dependendo do sistema operacional do usuário:

- No Mac OS: o diretório de armazenamento de um aplicativo varia de acordo com a versão do AIR:
  - **AIR 3.2 e anterior** - `<appData>/<appId>/Local Store/` onde `<appData>` é a "pasta de preferências" do usuário, normalmente: `/Users/<user>/Library/Preferences`
  - **AIR 3.3 e superior** - `<path>/Library/Application Support/<appID>/Local Store`, onde `<path>` é `/Users/<user>/Library/Containers/<bundle-id>/Data` (ambiente protegido) ou `/Users/<user>` (ao executar fora de um ambiente protegido)
- No Windows: o diretório de armazenamento do aplicativo é `<appData>\<appId>\Local Store\` onde `<appData>` é a "pasta especial" CSIDL_APPDATA do usuário, geralmente: `C:\Documents and Settings\<user>\Application Data`
- No Linux: `<appData>/<appID>/Local Store/` onde `<appData>` é `/home/<user>/.appdata`

Você pode acessar o diretório de armazenamento do aplicativo através da propriedade `air.File.applicationStorageDirectory`. Você pode acessar o respectivo conteúdo usando o método `resolvePath()` da classe File. Para obter detalhes, consulte"Trabalho com o sistema de arquivos" na página 673.

Atualização do Adobe AIR

Adobe AIR 1.0 e posterior

Quando o usuário instala um aplicativo do AIR que requer uma versão atualizada do tempo de execução, ele instala automaticamente o tempo de execução atualizado desejado.

Para atualizar o tempo de execução, o usuário deve ter privilégios administrativos no computador.

Atualização de aplicativos AIR

Adobe AIR 1.0 e posterior

O desenvolvimento e a implantação de atualizações de software são alguns dos maiores desafios de segurança que os aplicativos de código nativo enfrentam. A API do AIR oferece um mecanismo para melhorar isso: o método `Updater.update()` pode ser chamado na inicialização para verificar o local remoto de um arquivo AIR. Se a atualização for adequada, o arquivo AIR é baixado, instalado e o aplicativo reiniciado. Os desenvolvedores podem usar essa classe não apenas para oferecer novas funcionalidades, mas também para responder a vulnerabilidades potenciais de segurança.

A classe Updater pode ser usada somente para atualizar aplicativos distribuídos como arquivos do AIR. Aplicativos distribuídos como aplicativos nativos precisam usar os recursos de atualização do sistema operacional nativo, se houver.

Nota: *Os desenvolvedores podem especificar a versão do aplicativo definindo a propriedade versionNumber do arquivo descritor do aplicativo.*

Desinstalação do aplicativo do AIR

Adobe AIR 1.0 e posterior

Remover o aplicativo do AIR remove também todos os arquivos no diretório do aplicativo. No entanto, não remove todos os arquivos que o aplicativo possa ter gravado fora do diretório do aplicativo. Remover aplicativos AIR não reverte as alterações que o aplicativo fez nos arquivos fora do diretório do aplicativo.

Configurações de Registro do Windows para administradores

Adobe AIR 1.0 e posterior

No Windows, os administradores podem configurar o computador para impedir (ou permitir) a instalação de aplicativo do AIR e atualizações do tempo de execução. Essas configurações estão contidas no Registro do Windows na seguinte chave: HKLM\Software\Policies\Adobe\AIR. Elas incluem o seguinte:

Configuração do Registro	Descrição
AppInstallDisabled	Especifica se a instalação e a desinstalação do aplicativo do AIR é permitida. Defina como 0 para “permitido,” defina como 1 para “não permitido.”
UntrustedAppInstallDisabled	Especifica que será permitida a instalação de aplicativos AIR não confiáveis (aplicativos que não contêm um certificado confiável). Defina como 0 para “permitido,” defina como 1 para “não permitido.”
UpdateDisabled	Especifica se a atualização do tempo de execução é permitida como uma tarefa de plano de fundo ou como parte de uma instalação explícita. Defina como 0 para “permitido,” defina como 1 para “não permitido.”

Segurança HTML no Adobe AIR

Adobe AIR 1.0 e posterior

Este tópico descreve a arquitetura de segurança HTML do AIR e como usar iframes, quadros e a ponte da caixa de proteção para configurar aplicativos baseados em HTML e integrar conteúdo HTML com segurança em aplicativos baseados em SWF.

O tempo de execução aplica regras e oferece mecanismos para dominar possíveis vulnerabilidades de segurança em HTML e JavaScript. As mesmas regras são aplicadas se o aplicativo for gravado principalmente em JavaScript ou se você carregar conteúdo HTML e JavaScript em um aplicativo baseado em SWF. Conteúdo na caixa de proteção do aplicativo e a caixa de proteção de segurança que não seja de aplicativo têm privilégios distintos. Ao carregar o conteúdo em um iframe ou frame, o tempo de execução fornece um mecanismo de *ponte de caixa de proteção* seguro, que permite que o conteúdo frame ou iframe se comunique de maneira segura com o conteúdo na caixa de proteção de segurança do aplicativo.

O SDK do AIR oferece três classes para renderização de conteúdo HTML.

A classe HTMLLoader proporciona integração entre o código JavaScript e as APIs do AIR.

A classe StageWebView é uma classe de renderização de HTML e tem integração muito limitada com o aplicativo do AIR host. O conteúdo carregado pela classe StageWebView nunca é colocado na caixa de proteção de segurança do aplicativo e não pode acessar dados nem chamar funções no aplicativo do AIR host. Em plataformas de desktop, a classe StageWebView usa o mecanismo HTML incorporado do AIR, com base no Webkit, que também é usado pela classe HTMLLoader. Em plataformas móveis, a classe StageWebView usa o controle HTML fornecido pelo sistema operacional. Assim, em plataformas móveis, a classe StageWebView tem as mesmas considerações e vulnerabilidades de segurança do navegador da Web do sistema.

A classe TextField pode exibir strings de texto em HTML. Nenhum JavaScript pode ser executado, mas o texto pode incluir links e imagens carregadas externamente.

Para obter mais informações, consulte “Como evitar erros JavaScript relacionados à segurança” na página 1016.

Visão geral sobre a configuração de aplicativo baseado em HTML

Adobe AIR 1.0 e posterior

Os Frames e iframes oferecem uma estrutura conveniente para organizar conteúdo HTML no AIR. Os frames oferecem um meio de manter a persistência de dados e trabalhar de maneira segura com conteúdo remoto.

Como o HTML no AIR retém sua organização normal baseada em página, o ambiente HTML será totalmente atualizado, caso o frame superior do conteúdo HTML "navegue" para uma página diferente. Você pode usar frames e iframes para manter a persistência de dados no AIR, da mesma maneira que em um aplicativo da Web em execução em um navegador. Defina os objetos principais do aplicativo no frame superior e eles persistirão, desde que você não permita que o frame navegue para uma nova página. Use frames ou iframes filhos para carregar e exibir as partes transitórias do aplicativo. (Há diversas maneiras de manter a persistência de dados, que podem ser usadas além de ou em vez de frames. Isso inclui cookies, objetos locais compartilhados, armazenamento local de arquivos, depósito de arquivo criptografado e armazenamento de banco de dados local).

Como o HTML no AIR retém sua linha desfocada normal entre o código executável e os dados, o AIR coloca o conteúdo no quadro superior do ambiente HTML, na caixa de proteção do aplicativo. Depois do evento `load` da página, o AIR restringe quaisquer operações, como `eval()`, que possam converter uma sequência de caracteres de texto em um objeto executável. Essa restrição é aplicada mesmo quando o aplicativo não carrega conteúdo remoto. Para permitir que o conteúdo em HTML execute essas operações restritas, você deve usar frames ou iframes para colocar o conteúdo em uma caixa de proteção não aplicativo. (Executar conteúdo em um quadro filho de uma caixa de proteção pode ser necessário ao usar algumas estruturas do aplicativo JavaScript que dependem da função `eval()`.) Para obter uma lista completa das restrições JavaScript na caixa de proteção do aplicativo, consulte "Restrições de código de conteúdo em caixas de proteção distintas" na página 1123.

Como o HTML no AIR mantém a capacidade de carregar conteúdo remoto possivelmente inseguro, o AIR aplica a política de mesma origem que impede que o conteúdo de um domínio interaja com o conteúdo de outro domínio. Para permitir a interação entre o conteúdo do aplicativo e o conteúdo de outro domínio, você pode configurar uma ponte para servir como interface entre um frame pai e filho.

Configuração de relacionamento pai-filho de caixa de proteção

Adobe AIR 1.0 e posterior

O AIR adiciona os atributos `sandboxRoot` e `documentRoot` aos elementos frame e iframe de HTML. Esses atributos permitem tratar o conteúdo do aplicativo como se ele tivesse vindo de outro domínio:

Atributo	Descrição
sandboxRoot	A URL que deve ser usada para determinar a caixa de proteção e o domínio em que o conteúdo do frame deve ser colocado. Os esquemas de URL `file:`, `http:` ou `https:` devem ser usados.
documentRoot	A URL da qual o conteúdo do frame deve ser carregado. Os esquemas de URL `file:`, `app:` ou `app-storage:` devem ser usados.

O exemplo a seguir mapeia o conteúdo instalado no subdiretório da caixa de proteção do aplicativo a ser executado na caixa de proteção remota e o domínio www.exemplo.com:

```
<iframe
    src="ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

Configuração de ponte entre frames pai e filho em caixas de proteção ou domínios distintos

Adobe AIR 1.0 e posterior

O AIR adiciona as propriedades `childSandboxBridge` e `parentSandboxBridge` ao objeto `window` de qualquer frame filho. Essas propriedades permitem definir pontes para servir como interfaces entre um quadro pai e filho. Cada ponte segue em uma direção:

`childSandboxBridge` A propriedade `childSandboxBridge` permite que o frame filho exponha uma interface para o conteúdo do frame pai. Para expor uma interface, você define a propriedade `childSandbox` como função ou objeto no frame filho. Em seguida, você pode acessar o objeto ou a função do conteúdo no frame pai. O exemplo a seguir mostra como um script que está sendo executado em um frame filho pode expor um objeto contendo uma função e uma propriedade para o respectivo pai:

```
var interface = {};
interface.calculatePrice = function(){
    return .45 + 1.20;
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Se esse conteúdo filho estiver em um iframe com `id` de `"child"` atribuída, você poderá acessar a interface do conteúdo pai, lendo a propriedade `childSandboxBridge` do frame:

```
var childInterface = document.getElementById("child").childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces "1.65"
air.trace(childInterface.storeID)); //traces "abc"
```

`parentSandboxBridge` A propriedade `parentSandboxBridge` permite que o frame pai exponha uma interface para o conteúdo do frame filho. Para expor uma interface, você define a propriedade `parentSandbox` do frame filho como função ou objeto no frame pai. Em seguida, você pode acessar o objeto ou a função do conteúdo no frame filho. O exemplo a seguir mostra como um script em execução no frame pai pode expor um objeto contendo uma função save para o filho:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").parentSandboxBridge = interface;
```

Ao usar essa interface, o conteúdo no frame filho poderá salvar texto em um arquivo chamado save.txt. No entanto, ele não terá nenhum outro acesso ao sistema de arquivos. Em geral, o conteúdo do aplicativo deve expor a interface mais estreita possível para as outras caixas de proteção. O conteúdo filho poderá chamar a função save da seguinte maneira:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Se o conteúdo filho tentar definir uma propriedade do objeto `parentSandboxBridge`, o tempo de execução lançará uma exceção SecurityError. Se o conteúdo pai tentar definir uma propriedade do objeto `childSandboxBridge`, o tempo de execução lançará uma exceção SecurityError.

Restrições de código de conteúdo em caixas de proteção distintas

Adobe AIR 1.0 e posterior

Como discutido na introdução deste tópico, “Segurança HTML no Adobe AIR” na página 1121, o tempo de execução aplica regras e fornece mecanismos para dominar possíveis vulnerabilidades de segurança em HTML e JavaScript. Este tópico lista essas restrições. Se o código tentar chamar essas APIs restritas, o tempo de execução lançará um erro com a mensagem "Violação de segurança de tempo de execução do Adobe AIR para código JavaScript na caixa de proteção de segurança do aplicativo".

Para obter mais informações, consulte “Como evitar erros JavaScript relacionados à segurança” na página 1016.

Restrições sobre o uso da função eval() de JavaScript e técnicas semelhantes

Adobe AIR 1.0 e posterior

Para conteúdo HTML na caixa de proteção de segurança do aplicativo há limitações no uso das APIs que podem transformar dinamicamente as strings em código executável após o carregamento do código (após o evento `onload` do elemento `body` ter sido despachado e o término de execução da função do manipulador `onload`). Isso é para evitar que o aplicativo injete inadvertidamente (e execute) código de fontes "não aplicativo" (como domínios de rede potencialmente inseguros).

Por exemplo, se o aplicativo usa strings de dados de uma fonte remota para gravar na propriedade innerHTML de um elemento DOM, a string pode incluir o código (JavaScript) executável que pode executar operações inseguras. No entanto, enquanto o conteúdo estiver carregando, não há risco de inserir strings remotas no DOM.

Uma restrição está no uso da função `eval()` de JavaScript. Após o código na caixa de proteção do aplicativo tiver sido carregado e após o processamento do manipulador de eventos onload, você só poderá usar a função `eval()` de forma limitada. As seguintes regras se aplicam ao uso da função `eval()` *após* o código ter sido carregado da caixa de proteção de segurança do aplicativo:

- São permitidas expressões que envolvem literais. Por exemplo:

  ```
  eval("null");
  eval("3 + .14");
  eval("'foo'");
  ```

- As literais de objeto são permitidos da seguinte maneira:

  ```
  { prop1: val1, prop2: val2 }
  ```

- As opções setter/getters de literal de objeto são *proibidas*, conforme segue:

  ```
  { get prop1() { ... }, set prop1(v) { ... } }
  ```

- As literais de matriz são permitidas da seguinte maneira:

  ```
  [ val1, val2, val3 ]
  ```

- Expressões que envolvem leituras de propriedades são *proibidas*, conforme segue:

  ```
  a.b.c
  ```

- A chamada de função é *proibida*.
- Definições de função são *proibidas*.
- A configuração de qualquer propriedade é *proibida*.
- As literais de função são *proibidas*.

No entanto, enquanto o código estiver sendo carregado, antes do evento `onload` e durante a execução da função do manipulador de eventos `onload`, essas restrições não serão aplicadas ao conteúdo na caixa de proteção de segurança do aplicativo.

Por exemplo, após o código ser carregado, o código a seguir resultará no lançamento de uma exceção pelo tempo de execução.

```
eval("alert(44)");
eval("myFunction(44)");
eval("NativeApplication.applicationID");
```

O código gerado dinamicamente, como o que é feito durante a chamada da função `eval()`, representaria um risco à segurança se permitido na caixa de proteção do aplicativo. Por exemplo, um aplicativo pode executar inadvertidamente uma string carregada de um domínio de rede e essa string pode conter código mal-intencionado. Por exemplo, esse pode ser um código de exclusão ou alteração de arquivos no computador do usuário. Ou pode ser um código que informa o conteúdo de um arquivo local para um domínio de rede não confiável.

As formas de gerar código dinâmico são as seguintes:

- Chamando a função `eval()`.
- Uso de propriedades `innerHTML` ou funções DOM para inserir tags de script que carregam um script de fora do diretório do aplicativo.
- Uso de propriedades `innerHTML` ou funções DOM para inserir tags de scripts com código inline (em vez de carregar um script através do atributo `src`).
- Configuração do atributo `src` para que a tag de `script` carregue um arquivo JavaScript que está fora do diretório do aplicativo.
- Uso de esquema de URL `javascript` (como em `href="javascript:alert('Test')"`).
- Uso da função `setInterval()` ou `setTimout()` em que o primeiro parâmetro (que define a função para ser executada de forma assíncrona) é uma string (a ser avaliada) em vez de um nome de função (como em `setTimeout('x = 4', 1000)`).
- Chamada de `document.write()` ou `document.writeln()`.

O código na caixa de proteção de segurança do aplicativo só pode usar esses métodos enquanto o conteúdo estiver sendo carregado.

Essas restrições *não* impedem o uso de `eval()` com literais do objeto JSON. Isso permite que o conteúdo do aplicativo trabalhe com a biblioteca JavaScript JSON. No entanto, você não poderá usar código JSON sobrecarregado (com manipuladores de eventos)

Em outras estruturas Ajax e bibliotecas de código JavaScript, certifique-se de que o código na estrutura ou biblioteca funciona dentro dessas restrições em código gerado dinamicamente. Se não funcionarem, inclua algum conteúdo que use a estrutura ou biblioteca em uma caixa de proteção de segurança "não aplicativo". Consulte detalhes em “Restrições a JavaScript dentro do AIR” na página 1084 e “Script entre conteúdo de aplicativo e "não aplicativo"” na página 1133. O Adobe mantém uma lista das estruturas Ajax conhecidas por oferecer suporte à caixa de proteção de segurança do aplicativo, em http://www.adobe.com/products/air/develop/ajax/features/.

Ao contrário do conteúdo na caixa de proteção de segurança do aplicativo, o conteúdo JavaScript em uma caixa de proteção de segurança "não aplicativo" *pode* chamar a função `eval()` para executar código gerado dinamicamente a qualquer momento.

Restrições de acesso às APIs do AIR (para caixas de proteção que não são de aplicativo)

Adobe AIR 1.0 e posterior

O código JavaScript em uma caixa de proteção "não aplicativo" não tem acesso ao objeto `window.runtime` e, portanto, esse código não pode executar APIs do AIR. Se o conteúdo em uma caixa de proteção de segurança "não aplicativo" chamar o código a seguir, o aplicativo lançará uma exceção TypeError:

```
try {
    window.runtime.flash.system.NativeApplication.nativeApplication.exit();
}
catch (e)
{
    alert(e);
}
```

O tipo de exceção é TypeError (valor indefinido), porque o conteúdo na caixa de proteção "não aplicativo" não reconhece o objeto `window.runtime`, portanto, ele é considerado como valor indefinido.

Você pode expor a funcionalidade de tempo de execução ao conteúdo em uma caixa de proteção "não aplicativo" usando uma ponte de script. Para obter detalhes, consulte “Script entre conteúdo de aplicativo e "não aplicativo"” na página 1133.

Restrições no uso de chamadas XMLHttpRequest

Adobe AIR 1.0 e posterior

O conteúdo HTML na caixa de proteção de segurança do aplicativo não pode usar os métodos XMLHttpRequest síncronos para carregar dados de fora da caixa de proteção do aplicativo enquanto o conteúdo HTML estiver sendo carregado e durante o evento `onLoad`.

Por padrão, o conteúdo HTML nas caixas de proteção de segurança "não aplicativo" não têm permissão para usar o objeto XMLHttpRequest de JavaScript para carregar dados de outros domínios que não o domínio que está chamando a solicitação. A tag `frame` ou `iframe` pode incluir um atributo `allowcrosscomainxhr`. Configurar esse atributo com qualquer valor diferente de "null" permite que o conteúdo no frame ou iframe use o objeto XMLHttpRequest de Javascript para carregar dados de outros domínios que não sejam o domínio do código que está chamando a solicitação:

```
<iframe id="UI"
    src="http://example.com/ui.html"
    sandboxRoot="http://example.com/"
    allowcrossDomainxhr="true"
    documentRoot="app:/">
</iframe>
```

Para obter mais informações, consulte “Script entre conteúdos em domínios distintos” na página 1127.

Restrições no carregamento de elementos CSS, frame, iframe e img (para conteúdo em caixas de proteção que não são de aplicativo)

Adobe AIR 1.0 e posterior

O conteúdo HTML nas caixas de proteção de segurança remota (rede) pode carregar apenas conteúdo CSS, `frame`, `iframe` e `img` de caixas de proteção remotas (de URLs de rede).

O conteúdo HTML nas caixas de proteção local com sistema de arquivos, local com rede ou local confiável só podem carregar conteúdo CSS, frame, iframe e `img` de caixas de proteção locais (e não de caixas de proteção de aplicativo ou remotas).

Restrições na chamada do método window.open() de JavaScript

Adobe AIR 1.0 e posterior

Se a janela criada através de uma chamada para o método `window.open()` de JavaScript exibir conteúdo de uma caixa de proteção de segurança "não aplicativo", o título da janela iniciará com o título da janela principal (inicializada), seguido pelo caractere dois pontos. Você não pode usar código para remover da tela essa parte do título da janela.

O conteúdo nas caixas de proteção de segurança "não aplicativo" só consegue chamar com êxito o método `window.open()` de JavaScript em resposta a um evento acionado pelo mouse do usuário ou pela interação do teclado. Isso evita que o conteúdo "não aplicativo" crie janelas que possam ser usadas de maneira enganosa (por exemplo, em ataques de phishing). Além disso, o manipulador de eventos do evento de mouse ou teclado não pode configurar o método `window.open()` para ser executado após um atraso (por exemplo, chamando a função `setTimeout()`).

O conteúdo nas caixas de proteção remotas (rede) só podem usar o método `window.open()` para abrir o conteúdo nas caixas de proteção de rede remota. Ele não pode usar o método `window.open()` para abrir o conteúdo das caixas de proteção locais ou do aplicativo.

O conteúdo nas caixas de proteção local com sistema de arquivos, local com rede ou local confiável (consulte "Caixas de proteção de segurança" na página 1081) só pode usar o método `window.open()` para abrir o conteúdo nas caixas de proteção locais. Ele não pode usar o método `window.open()` para abrir o conteúdo das caixas de proteção remotas ou do aplicativo.

Erros na chamada de código restrito

Adobe AIR 1.0 e posterior

Se você chamar um código com uso restrito em uma caixa de proteção, devido a essas restrições de segurança, o tempo de execução despachará um erro de JavaScript: "Violação de segurança de tempo de execução do Adobe AIR para código JavaScript na caixa de proteção de segurança do aplicativo."

Para obter mais informações, consulte "Como evitar erros JavaScript relacionados à segurança" na página 1016.

Proteção da caixa de proteção ao carregar conteúdo HTML de uma sequência de caracteres

Adobe AIR 1.0 e posterior

O método `loadString()` da classe HTMLLoader permite criar conteúdo HTML no tempo de execução. No entanto, os dados usados como conteúdo HTML podem ser corrompidos se forem carregados de uma fonte de Internet insegura. Por esse motivo, por padrão, o HTML criado com o método `loadString()` não é colocado na caixa de proteção do aplicativo e não tem acesso às APIs do AIR. Entretanto, você pode definir a propriedade `placeLoadStringContentInApplicationSandbox` de um objeto HTMLLoader como true para inserir o HTML criado usando o método `loadString()` na caixa de proteção do aplicativo. Para obter mais informações, consulte "Carregamento de conteúdo HTML de uma sequência de caracteres" na página 1015.

Script entre conteúdos em domínios distintos

Adobe AIR 1.0 e posterior

Aplicativos AIR recebem privilégios especiais quando são instalados. É fundamental que os mesmos privilégios não vazem para outro conteúdo, incluindo arquivos remotos e arquivos locais que não fazem parte do aplicativo.

Sobre a ponte de caixa de proteção do AIR

Adobe AIR 1.0 e posterior

Normalmente, o conteúdo de outros domínios não pode chamar scripts de outros domínios. Para proteger os aplicativos AIR de vazamentos acidentais de informações privilegiadas ou de controle, as restrições a seguir são colocadas no conteúdo na caixa de proteção de segurança do `aplicativo` (conteúdo instalado com o aplicativo):

- O código na caixa de proteção de segurança do aplicativo não pode permitir que outras caixas de proteção chamem o método `Security.allowDomain()`. Chamar esse método de uma caixa de proteção de segurança do aplicativo gerará um erro.
- Evita-se importar conteúdo "não aplicativo" da caixa de proteção do aplicativo, definindo a propriedade `LoaderContext.securityDomain` ou `LoaderContext.applicationDomain`.

Ainda há casos em que o aplicativo principal do AIR requer que o conteúdo de um domínio remoto tenha acesso controlado a scripts no aplicativo principal do AIR ou vice-versa. Para fazer isso, o tempo de execução oferece o mecanismo *ponte de caixa de proteção*, que serve como gateway entre as duas caixas de proteção. A ponte de caixa de proteção pode oferecer interação explícita entre as caixas de proteção de segurança do aplicativo e remota.

A ponte de caixa de proteção expõe dois objetos que os scripts, carregados e em carregamento, podem acessar:

- O objeto `parentSandboxBridge` permite carregar propriedades e funções de exposição de conteúdo para scripts no conteúdo carregado.
- O objeto `childSandboxBridge` permite que o conteúdo carregado exponha propriedades e funções para scripts no conteúdo que está em carregamento.

Os objetos expostos através da ponte de caixa de proteção são passadas por valor e não por referência. Todos os dados são serializados. Isso significa que os objetos expostos de um lado da ponte não podem ser definidos pelo outro lado da ponte e que os objetos expostos são todos sem categoria. Além disso, você pode expor objetos e funções simples, você não pode expor objetos complexos.

Se o conteúdo filho tentar definir um objeto para o objeto parentSandboxBridge, o tempo de execução lançará uma exceção SecurityError. Da mesma forma, se o conteúdo pai tentar definir um objeto para o objeto childSandboxBridge, o tempo de execução lançará uma exceção SecurityError.

Exemplo de ponte de caixa de proteção (SWF)

Adobe AIR 1.0 e posterior

Suponhamos que um aplicativo de armazenamento de música do AIR deseja permitir que arquivos remotos SWF transmitam preços de álbuns, mas não deseja que o arquivo remoto SWF divulgue se é preço de venda. Para fazer isso, a classe StoreAPI oferece um método para adquirir o preço, mas ocultar o preço de venda. Em seguida, é atribuída uma ocorrência dessa classe StoreAPI à propriedade `parentSandboxBridge` do objeto LoaderInfo do objeto Loader que carrega o SWF remoto.

A seguir está o código para armazenamento de música do AIR:

```
<?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>
```

O objeto StoreAPI chama o aplicativo principal para recuperar o preço regular do álbum, mas retorna "Não disponível" quando o método `getSaleAlbumPrice()` é chamado. O código a seguir define a classe StoreAPI:

```
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();
    }
}
```

O seguinte código representa um exemplo de um arquivo PriceQuoter SWF que informa o preço da loja, mas não pode informar o preço de venda:

```
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());
        }
    }
}
```

Exemplo de ponte de caixa de proteção (HTML)

Adobe AIR 1.0 e posterior

No conteúdo HTML, as propriedades `parentSandboxBridge` e `childSandboxBridge` são adicionados ao objeto de janela de JavaScript de um documento filho. Para obter um exemplo sobre como definir funções de ponte em conteúdo HTML, consulte "Configuração de interface de ponte de caixa de proteção" na página 1034.

Limitação de exposição de API

Adobe AIR 1.0 e posterior

Ao expor pontes de caixa de proteção, é importante expor APIs de alto nível que limitem o grau em que podem ser abusadas. Lembre-se de que o conteúdo que chama a implementação da ponte pode estar comprometido (por exemplo, através de uma injeção de código). Portanto, por exemplo, expor um método `readFile(path:String)` (que lê o conteúdo de um arquivo arbitrário) através de uma ponte, estará vulnerável a abusos. Talvez seja melhor expor uma API `readApplicationSetting()` que não siga um caminho e leia um arquivo específico. A abordagem mais semântica limita o dano que o aplicativo pode causar, já que parte dele está comprometida.

Mais tópicos da Ajuda

"Conteúdo cross-scripting em caixas de proteção de segurança distintas" na página 1033

"A caixa de proteção do aplicativo do AIR" na página 1082

Gravação em disco

Adobe AIR 1.0 e posterior

Os aplicativos em execução em um navegador da Web só têm interação limitada com o sistema de arquivos local do usuário. Os navegadores da Web implementam políticas de segurança que garantem que o computador do usuário não pode ser comprometido como resultado do carregamento de conteúdo da Web. Por exemplo, os arquivos SWF executados por meio do Flash Player em um navegador não podem interagir diretamente com os arquivos existentes no computador do usuário. Os objetos compartilhados e os cookies podem ser gravados no computador do usuário com a finalidade de manter as preferências do usuário e outros dados, mas esse é o limite de interação do sistema de arquivos. Como os aplicativos AIR são instalados de forma nativa, eles têm contratos de segurança diferentes, um dos quais inclui a capacidade de leitura e gravação no sistema de arquivos local.

Essa liberdade resulta em bastante responsabilidade para os desenvolvedores A falta de segurança acidental do aplicativo coloca em risco não apenas a funcionalidade do aplicativo, mas também a integridade do computador do usuário. Por esse motivo, os desenvolvedores devem ler as "Práticas recomendadas de segurança para desenvolvedores" na página 1134.

Os desenvolvedores do AIR podem acessar e gravar arquivos no sistema de arquivos local usando diversas convenções de esquema de URL:

esquema de URL	Descrição
app:/	Um alias para o diretório do aplicativo. Aos arquivos acessados desse caminho são atribuídas caixas de proteção do aplicativo e eles têm todos os privilégios concedidos pelo tempo de execução.
app-storage:/	Um alias para o diretório de armazenamento local, padronizado pelo tempo de execução. Aos arquivos acessados desse caminho é atribuída uma caixa de proteção "não aplicativo".
file:///	Um alias que representa a raiz do disco rígido do usuário. Ao arquivo acessado desse caminho é atribuída uma caixa de proteção do aplicativo, se o arquivo estiver no diretório do aplicativo e, caso contrário, uma caixa de proteção "não aplicativo".

Nota: *Os aplicativos AIR não podem modificar o conteúdo que usa o app: esquema de URL. Além disso, o diretório do aplicativo pode ser lido somente devido às configurações do administrador.*

A menos que haja restrições do administrador para o computador do usuário, os aplicativos AIR têm privilégio de gravação em qualquer local no disco rígido do usuário, Recomenda-se que os desenvolvedores usem o caminho `app-storage:/` para armazenamento local em relação ao aplicativo. Os arquivos gravados em `app-storage:/` de um aplicativo são colocados em um local padrão:

- No Mac OS: o diretório de armazenamento do aplicativo é `<appData>/<appId>/Local Store/` onde `<appData>` é a pasta Preferências do usuário, que é normalmente `/Users/<user>/Library/Preferences`
- No Windows: o diretório de armazenamento do aplicativo é `<appData>\<appId>\Local Store\` onde `<appData>` é a pasta especial CSIDL_APPDATA do usuário, que é normalmente `C:\Documents and Settings\<userName>\Application Data`
- No Linux: `<appData>/<appID>/Local Store/` onde `<appData>` é `/home/<user>/.appdata`

Se o aplicativo for desenvolvido para interagir com os arquivos existentes no sistema de arquivos do usuário, certifique-se de ler as "Práticas recomendadas de segurança para desenvolvedores" na página 1134.

Trabalho seguro com conteúdo não confiável

Adobe AIR 1.0 e posterior

O conteúdo não atribuído à caixa de proteção do aplicativo pode oferecer funcionalidade adicional de script ao aplicativo, mas somente se ele atender aos critérios de segurança do tempo de execução. Este tópico explica o contrato de segurança do AIR com conteúdo "não aplicativo".

Security.allowDomain()

Adobe AIR 1.0 e posterior

Os aplicativos AIR restringem o acesso de script de conteúdo "não aplicativo" com mais rigor do que o plug-in de navegador do Flash Player restringe o acesso de script de conteúdo não confiável. Por exemplo, no Flash Player, no navegador, quando um arquivo SWF atribuído à caixa de proteção `local-trusted` chama o método `System.allowDomain()`, todos os SWF carregados do domínio especificado receberão acesso de script. Não é permitida a abordagem análoga do conteúdo do `aplicativo` nos aplicativos AIR, já que ela concederia acesso excesso não razoável ao arquivo "não aplicativo" no sistema de arquivos do usuário. Os arquivos remotos não podem acessar diretamente a caixa de proteção do aplicativo, independentemente das chamadas para o método`Security.allowDomain()`.

Script entre conteúdo de aplicativo e "não aplicativo"

Adobe AIR 1.0 e posterior

Os aplicativos AIR que fazem script entre conteúdo aplicativo e "não aplicativo" têm organizações de segurança mais complexas. Os arquivos que não estão na caixa de proteção do aplicativo só têm permissão para acessar propriedades e métodos de aplicativos na caixa de proteção do aplicativo por meio do uso de uma ponte de caixa de proteção. A ponte da caixa de proteção atua como gateway entre o conteúdo do aplicativo e "não aplicativo", oferecendo interação explícita entre os dois arquivos. Quando usadas corretamente, as pontes de caixa de proteção oferecem uma camada extra de segurança, impedindo o conteúdo "não aplicativo" de acessar as referências de objeto que fazem parte do conteúdo do aplicativo.

O benefício das pontes de caixa de proteção é ilustrado melhor através do exemplo. Suponhamos que um aplicativo de armazenamento de música do AIR deseja fornecer uma API para anunciantes que desejam criar seus próprios arquivos SWF, com o qual o aplicativo de armazenamento poderá se comunicar em seguida. A loja deseja fornecer aos anunciantes métodos de pesquisa de artistas e CDs, mas também deseja isolar alguns métodos e propriedades do arquivo SWF terceirizado, por motivos de segurança.

A ponte de caixa de proteção pode propiciar essa funcionalidade. Por padrão, o conteúdo carregado externamente em um aplicativo do AIR em tempo de execução não tem acesso a nenhum método ou propriedade no aplicativo principal. Com a implementação da ponte de caixa de proteção personalizada, o desenvolvedor pode fornecer serviços ao conteúdo remoto sem expor esses métodos ou propriedades. Considere a ponte de caixa de proteção como um caminho entre o conteúdo confiável e o não confiável, oferecendo comunicação entre o conteúdo carregado e o do carregador sem expor as referências do objeto.

Para obter mais informações sobre como usar pontes de caixa de proteção de forma segura, consulte “Script entre conteúdos em domínios distintos” na página 1127.

Proteção contra conteúdo SWF inseguro gerado dinamicamente.

Adobe AIR 1.0 e posterior

O método `Loader.loadBytes()` oferece uma maneira de o aplicativo gerar conteúdo SWF de uma matriz de bytes. No entanto, ataques de injeção em dados carregados de fontes remotas podem causar dano grave ao carregar o conteúdo. Isso é particularmente verdadeiro ao carregar dados na caixa de proteção do aplicativo, onde o conteúdo SWF gerado pode acessar o conjunto completo de APIs do AIR.

Há usos válidos para a utilização do método `loadBytes()` sem gerar um código SWF executável. Você pode usar o método `loadBytes()` para gerar dados de imagem para controlar o tempo de exibição de imagem, por exemplo. Há também usos válidos que *dependem* do código de execução, como a criação dinâmica de conteúdo SWF para reprodução de áudio. No AIR, por padrão, o método `loadBytes()` *não* permite carregar conteúdo SWF, ele só permite carregar conteúdo de imagem. No AIR, a propriedade `loaderContext` do método `loadBytes()` tem uma propriedade `allowLoadBytesCodeExecution`, que você pode definir como `true` para permitir explicitamente que o aplicativo use `loadBytes()` para carregar o conteúdo SWF executável. O código a seguir mostra como usar esse recurso:

```
var loader:Loader = new Loader();
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.allowLoadBytesCodeExecution = true;
loader.loadBytes(bytes, loaderContext);
```

Se você chamar `loadBytes()` para carregar conteúdo SWF e a propriedade `allowLoadBytesCodeExecution` do objeto LoaderContext estiver definida como `false` (o padrão), o objeto Loader lançará uma exceção SecurityError.

Nota: *Em um lançamento futuro do Adobe AIR, essa API pode ser alterada. Quando isso ocorre, talvez seja necessário recompilar o conteúdo que usa a propriedade* `allowLoadBytesCodeExecution` *da classe LoaderContext.*

Práticas recomendadas de segurança para desenvolvedores

Adobe AIR 1.0 e posterior

Embora os aplicativos AIR sejam criados usando tecnologias da Web, é importante para os desenvolvedores observar que eles não trabalham na caixa de proteção de segurança do navegador. Isso significa que é possível criar aplicativos AIR que podem danificar o sistema local de maneira intencional ou não intencional. O AIR tenta minimizar esse risco, mais ainda há formas em que as vulnerabilidades podem ser introduzidas. Este tópico cobre faltas de segurança potenciais importantes.

Risco de importar arquivos para a caixa de proteção de segurança do aplicativo

Adobe AIR 1.0 e posterior

Os arquivos existentes no diretório do aplicativo estão atribuídos à caixa de proteção do aplicativo e têm privilégios completos de tempo de execução. Recomenda-se aos aplicativos que gravam no sistema de arquivos local gravar no `app-storage:/`. Esse diretório está separado dos arquivos do aplicativo no computador do usuário, por isso os arquivos não estão atribuídos na caixa de proteção do aplicativo e representam risco reduzido de segurança. Recomenda-se aos desenvolvedores que considerem o seguinte:

- Somente incluir um arquivo em um arquivo do AIR (no aplicativo instalado) se for necessário.
- Somente incluir um arquivo em um arquivo do AIR (no aplicativo instalado) se o respectivo comportamento for completamente compreensível e confiável.
- Não grave nem modifique conteúdo no diretório do aplicativo. O tempo de execução impede que aplicativos gravem ou modifiquem arquivos e diretórios que usam o esquema de URL `app:/`, lançando uma exceção SecurityError.
- Não use dados de uma fonte de rede como parâmetros de métodos da API do AIR que possam conduzir a execução de código. Isso inclui o uso do método `Loader.loadBytes()` e a função `eval()` de JavaScript.

Risco de uso de fonte externa para determinar caminhos

Adobe AIR 1.0 e posterior

O aplicativo do AIR pode estar com o uso de dados ou conteúdo externo comprometido. Por esse motivo, tenha cuidado especial ao usar dados da rede ou do sistema de arquivos. O ônus da confiança, depende em última análise do desenvolvedor e das conexões de rede que eles fazem, mas carregar dados externos é inerentemente arriscado e não deve ser usado em inserções de operações confidenciais. Recomenda-se aos desenvolvedores o seguinte:

- Usar dados de uma fonte de rede para determinar o nome de arquivo
- Usar dados de uma fonte de rede para construir uma URL que o aplicativo use para enviar informações particulares

O risco de usar, armazenar ou transmitir credenciais não seguras

Adobe AIR 1.0 e posterior

Armazenar credenciais do usuário no sistema de arquivos local do usuário introduz inerentemente o risco de que essas credenciais possam estar comprometidas. Recomenda-se aos desenvolvedores que considerem o seguinte:

- Se credenciais devem ser armazenadas localmente, criptografe as credenciais ao gravá-las no sistema de arquivos local. O tempo de execução oferece um armazenamento criptografado exclusivo para cada aplicativo instalado, através da classe EncryptedLocalStore. Para ver detalhes, consulte "Armazenamento local criptografado" na página 733.
- Não transmita credenciais de usuário não-criptografadas, exceto se a fonte for confiável e a transmissão utilizar HTTPS: ou protocolos Transport Layer Security (TLS).
- Nunca especifique uma senha padrão na criação de credencial: deixe que os usuários criem suas próprias senhas. Usuários que mantêm o padrão inalterado expõem as respectivas credenciais ao invasor que já conhece a senha padrão.

Risco de um ataque de downgrade

Adobe AIR 1.0 e posterior

Durante a instalação do aplicativo, o tempo de execução certifica-se de que não haja uma versão do aplicativo instalada atualmente. Se o aplicativo já estiver instalado, o tempo de execução compara a string da versão em relação à versão que está sendo instalada. Se essa string for diferente, o usuário poderá optar por atualizar a instalação. O tempo de execução não garante que a versão instalada recentemente seja mais recente que a versão anterior, apenas que seja diferente. O invasor pode distribuir uma versão anterior para o usuário, para contornar uma falha de segurança. Por esse motivo, recomenda-se que o desenvolvedor verifique as versões quando o aplicativo for executado. É uma boa ideia fazer com que os aplicativos verifiquem a rede em busca de atualizações necessárias. Dessa maneira, mesmo se um invasor fizer com que o usuário execute uma versão anterior, essa versão anterior irá reconhecer que é necessário atualizar. Além disso, usar um esquema de versão bem-definido no aplicativo torna mais difícil enganar os usuários para que instalem uma versão desatualizada.

Assinatura de código

Adobe AIR 1.0 e posterior

Todos os arquivos do instalador do AIR são obrigatórios para assinatura de código. A assinatura de código é um processo criptográfico para confirmar que a origem especificada do software é precisa. Os aplicativos AIR podem ser assinados por um certificado emitido por uma autoridade de certificação (CA) externa ou por um certificado autoassinado que você tenha criado. Recomenda-se fortemente um certificado comercial de uma CA bem conhecida que ofereça segurança a seus usuários de que estão instalando o aplicativo original e não uma falsificação. No entanto, certificados autoassinados podem ser criados com `adt` do SDK ou com o Flash, Flash Builder ou outro aplicativo que utilize `adt` para geração de certificados. Os certificados autoassinados não garantem a legitimidade do aplicativo que estiver sendo instalado e devem ser usados apenas para testar um aplicativo antes da distribuição ao público.

Segurança em dispositivos Android

Adobe AIR 2.5 e posterior

No Android, como em todos os dispositivos de computação, o AIR se conforma com o modelo de segurança nativo. Ao mesmo tempo, o AIR mantém suas próprias regras de segurança, que se destinam a facilitar aos desenvolvedores a programação de aplicativos seguros conectados à Internet.

Visto que aplicativos AIR no Android usam o formato de pacote do Android, a instalação se ajusta ao modelo de segurança do Android. O instalador de aplicativo do AIR não é usado.

O modelo de segurança do Android tem três aspectos principais:

- Permissões
- Assinaturas do aplicativo
- IDs de usuário do aplicativo

Permissões do Android

Muitos recursos do Android são protegidos pelo mecanismo de permissões do sistema operacional. A fim de usar um recurso protegido, o descritor do aplicativo do AIR precisa declarar que o aplicativo requer a permissão necessária. Quando um usuário tenta instalar o aplicativo, o sistema operacional Android exibe todas as permissões solicitadas ao usuário antes de continuar com a instalação.

A maioria dos aplicativos AIR precisam especificar as permissões do Android no descritor do aplicativo. Por padrão, nenhuma permissão é incluída. As seguintes permissões são necessárias para recursos protegidos do Android expostos pelo tempo de execução do AIR:

ACCESS_COARSE_LOCATION Permite ao aplicativo acessar dados de local de rede WIFI e celular por meio da classe Geolocation.

ACCESS_FINE_LOCATION Permite ao aplicativo acessar dados GPS por meio da classe Geolocation.

ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE Permite ao aplicativo acessar informações de rede por meio da classe NetworkInfo.

CAMERA Permite ao aplicativo acessar a câmera.

INTERNET Permite ao aplicativo fazer solicitações de rede, além de permitir a depuração remota.

READ_PHONE_STATE Permite ao tempo de execução do AIR silenciar o áudio quando uma chamada é recebida.

RECORD_AUDIO Permite ao aplicativo acessar o microfone.

WAKE_LOCK e DISABLE_KEYGUARD Permite ao aplicativo impedir que o dispositivo entre no modo de suspensão por meio do uso das configurações da classe SystemIdleMode.

WRITE_EXTERNAL_STORAGE Permite ao aplicativo gravar no cartão de memória externo no dispositivo.

Assinaturas do aplicativo

Todos os pacotes de aplicativos criados para a plataforma Android precisam ser assinados. Visto que os aplicativos AIR no Android são empacotados no formato APK nativo do Android, eles são assinados de acordo com as convenções do Android, e não com as convenções do AIR. Embora o Android e o AIR usem a assinatura de código de forma semelhante, há três diferenças significativas:

- No Android, a assinatura verifica se o desenvolvedor está de posse da chave privada, mas não é usada para verificar a identidade do desenvolvedor.

- Para aplicativos enviados ao Android Market, o certificado precisa ser válido por pelo menos 25 anos.
- O Android não oferece suporte a migração da assinatura do pacote para outro certificado. Se uma atualização for assinada por um certificado diferente, os usuários precisam instalar o aplicativo original para poderem instalar o aplicativo atualizado.
- Dois aplicativos assinados com o mesmo certificado podem especificar uma ID compartilhada que permita a eles acessar o cache e os arquivos de dados um do outro. (Esse compartilhamento não é facilitado pelo AIR.)

IDs de usuário do aplicativo

O Android usa um núcleo do Linux. A cada aplicativo instalado é atribuída uma ID de usuário do tipo do Linux que determina suas permissões para operações como acesso a arquivos. Os arquivos no aplicativo, o armazenamento do aplicativo e os diretórios temporários são protegidos contra o acesso por permissões do sistema de arquivos. Arquivos programados para armazenamento externo (em outras palavras, o cartão SD) podem ser lidos, modificados ou excluídos por outros aplicativos ou pelo usuário quando o cartão SD é montado como dispositivo de armazenamento em massa em um computador.

Cookies recebidos com solicitações da Internet não são compartilhados entre aplicativos AIR.

Privacidade da imagem do fundo

Quando um usuário envia um aplicativo para o fundo, algumas versões do Android capturam a tela que ele usa como uma miniatura na lista de aplicativos recentes. Essa captura de tela é armazenada na memória do dispositivo e pode ser acessada por um invasor que tenha o controle físico do dispositivo.

Se seu aplicativo exibir informações confidenciais, você deve se proteger contra a captura dessas informações por meio da captura de tela da imagem de fundo. O evento `deactivate` despachado pelo objeto NativeApplication sinaliza que um aplicativo está prestes a alternar para a imagem de fundo. Use esse evento para limpar ou ocultar quaisquer informações confidenciais.

Mais tópicos da Ajuda

Android: segurança e permissões

Dados criptografados no Android

Aplicativos AIR no Android podem usar as opções de criptografia disponíveis no banco de dados SQL incorporado para salvar dados criptografados. Para otimizar a segurança, posicione a chave de criptografia sobre uma senha digitada pelo usuário que é digitada sempre que o aplicativo está em execução. Uma chave de criptografia ou uma senha armazenada localmente é difícil ou impossível de ser "ocultada" de um hacker que tenha acesso aos arquivos do aplicativo. Caso o hacker consiga salvar a chave , então a criptografia dos dados não oferecerá a proteção adicional além do sistema de arquivos de segurança baseado no ID do usuário fornecido pelo sistema Android.

A classe EncryptedLocalStore pode ser usada para salvar dados, mas esses dados não serão criptografados nos dispositivos do Android. Em vez disso, o modelo de segurança do Android confia no ID de usuário do aplicativo para proteger os dados de outros aplicativos. Os aplicativos que utilizam um ID de usuário compartilhado e que são assinados com o mesmo certificado de assinatura de código utilizam o mesmo armazenamento local criptografado.

Importante: *Em um telefone enraizado, qualquer aplicativo em execução com privilégios de raiz pode acessar os arquivos de qualquer outro aplicativo. Dessa forma, os dados armazenados que utilizam o armazenamento local criptografado não estão seguros em um dispositivo enraizado.*

Segurança em dispositivos iOS

No iOS, o AIR se adapta ao modelo de segurança nativo. Ao mesmo tempo, o AIR mantém suas próprias regras de segurança, que se destinam a facilitar aos desenvolvedores a programação de aplicativos seguros conectados à Internet.

Visto que os aplicativos do AIR no Android usam o formato de pacote do Android, a instalação se ajusta ao modelo de segurança do iOS. O instalador de aplicativo do AIR não é usado. Além disso, um tempo de execução separado do AIR não é usado em dispositivos iOS. Todos aplicativos do AIR contêm todos os códigos necessários para funcionar.

Assinaturas do aplicativo

Todos os pacotes de aplicativos criados para a plataforma iOS precisam ser assinados. Uma vez que os aplicativos do AIR no iOS são empacotados no formato IPA do iOS nativo, eles são assinados de acordo com os requisitos do iOS, não de acordo com os requisitos do AIR. Embora o iOS e o AIR usem a assinatura de código de forma semelhante, há três diferenças significativas:

- No iOS, o certificado usado para assinar um aplicativo deve ser emitido pela Apple; os certificados de outras autoridades não podem ser usados.
- No iOS, os certificados de distribuição emitidos pela Apple normalmente são válidos por um ano.

Privacidade da imagem do fundo

Quando um usuário alterna um aplicativo para a imagem de fundo no iOS, o sistema operacional faz uma captura de tela e a usa para animar a transição. Essa captura de tela é armazenada na memória do dispositivo e pode ser acessada por um invasor que tenha o controle físico do dispositivo.

Se seu aplicativo exibir informações confidenciais, você deve se proteger contra a captura dessas informações por meio da captura de tela da imagem de fundo. O evento `deactivate` despachado pelo objeto NativeApplication sinaliza que um aplicativo está prestes a alternar para a imagem de fundo. Use esse evento para limpar ou ocultar quaisquer informações confidenciais.

Capítulo 65: Como Usar Exemplos do ActionScript

A execução de um exemplo de código do ActionScript 3.0 é uma das melhores maneiras para entender como determinadas classes e métodos funcionam. Você pode usar os exemplos de maneiras diferentes, dependendo dos dispositivos que estiver usando ou direcionando

Computadores que operam com Flash Professional ou Flash Builder Consulte "Execução de exemplos do ActionScript 3.0 no Flash Professional" na página 1141 ou "Execução de exemplos do ActionScript 3.0 no Flash Builder" na página 1142 para obter informações sobre como usar esses ambientes de desenvolvimento ao executar exemplos do ActionScript 3.0. Use instruções de rastreio e outras ferramentas de depuração para compreender melhor o funcionamento de um exemplo de código.

Dispositivos móveis Você pode executar os exemplos de código do ActionScript 3.0 em dispositivos móveis com suporte a Flash Player 10.1 e versões posteriores. Consulte "Execução de exemplos do ActionScript 3.0 em dispositivos móveis" na página 1144. Você também pode executar esses exemplos em seu computador, usando o Flash Professional ou o Flash Builder.

Aparelho de TV Embora não seja possível executar esses exemplos em aparelhos de TV, você ainda pode compreender os exemplos executando-os em seu computador. Consulte Plataforma Flash para TV no site Adobe Developer Connection para obter informações sobre como desenvolver aplicativos de aparelhos de TV.

Tipos de exemplos

Os tipos de exemplos de código do ActionScript 3.0 são:

- Exemplos de código snippet (encontrados no conjunto de documentação do ActionScript 3.0)
- Exemplos baseados em classe (encontrados principalmente em Referência do ActionScript 3.0 para Adobe Flash Platform)
- Exemplos práticos que contém vários arquivos de origem (baixe os arquivos ZIP de origem de www.adobe.com/go/learn_programmingAS3samples_flash_br)

Exemplos de snippets de código

Um exemplo de snippet de código tem o seguinte aspecto:

```
var x:int = 5;
trace(x); // 5
```

Os snippets de código contêm apenas o código suficiente para demonstrar uma única ideia. Normalmente, eles não contêm instruções de pacote ou classe.

Exemplos com base em classes

Muitos exemplos mostram o código-fonte de uma classe ActionScript completa. Um exemplo com base em classe tem o seguinte aspecto:

```
package {
    public class Example1 {
        public function Example1():void {
            var x:int = 5;
            trace(x); //5
        }
    }
}
```

O código de um exemplo com base em classe contém uma instrução de pacote, uma declaração de classe e uma função construtora.

Exemplos práticos que contêm mais de um arquivo de origem

Muitos tópicos do Guia do desenvolvedor do ActionScript 3.0 terminam com exemplos práticos que mostram como usar determinados recursos do ActionScript em um contexto prático e real. Normalmente, esses exemplos contêm vários arquivos, inclusive:

- Um ou mais arquivos de origem do ActionScript
- Um arquivo .FLA para ser usado com o Flash Professional
- Um ou mais arquivos MXML para ser usado com o Flash Builder
- Arquivos de dados, arquivos de imagem, arquivos de som ou outros recursos usados pelo aplicativo de exemplo (opcional).

Normalmente, os exemplos práticos são distribuídos como arquivos ZIP.

Lista de exemplos do Manual do Desenvolvedor do arquivo ZIP

O arquivo ZIP para Flash Professional CS5 e Flex 4 (baixe em www.adobe.com/go/learn_programmingAS3samples_flash_br) contém os seguintes exemplos:

- AlarmClock (“Exemplo de tratamento de evento: relógio de alarme” na página 142)
- AlgorithmicVisualGenerator (“Exemplo de API de desenho: gerador visual algorítmico” na página 237)
- ASCIIArt (“Exemplos de sequências de caracteres: arte ASCII” na página 19)
- CapabilitiesExplorer (“Exemplo de recursos: detecção de capacidades do sistema” na página 901)
- CustomErrors (“Exemplo de tratamento de erros: aplicativo CustomErrors” na página 70)
- DisplayObjectTransformer (“Exemplo de geometria: aplicação de uma transformação de matriz em um objeto de exibição” na página 221)
- FilterWorkbench (“Exemplo de filtragem de objetos de exibição: bancada do filtro” na página 300)
- GlobalStockTicker (“Exemplo: Internacionalização de um aplicativo de acionamento de ações” na página 986)
- IntrovertIM_HTML (“Exemplo de API externa: comunicação entre ActionScript e JavaScript em um navegador da Web” na página 877)
- NewsLayout (“Exemplo de TextField: formatação de texto estilo jornal” na página 398)
- PlayList (“Exemplo de matriz: PlayList” na página 48)
- PodcastPlayer (“Exemplo de som: Podcast Player” na página 480)
- ProjectionDragger (“Exemplo: Projeção em perspectiva” na página 367)
- ReorderByZ (“Uso de objetos Matrix3D para reordenar a exibição” na página 372)
- RSSViewer (“Exemplo XML no ActionScript: carregamento de dados RSS da Web” na página 114)

- RuntimeAssetsExplorer (“Exemplo de clipe de filme: RuntimeAssetsExplorer” na página 340)
- SimpleClock (“Exemplo de data e hora: relógio analógico simples” na página 6)
- SpinningMoon (“Exemplo de bitmap: lua giratória animada” na página 260)
- SpriteArranger (“Exemplo de objeto de exibição: SpriteArranger” na página 207)
- TelnetSocket (“Exemplo de soquete TCP: construção de um cliente Telnet” na página 825)
- VideoJukebox (“Exemplo de vídeo: jukebox de vídeo” na página 522)
- WikiEditor (“Exemplo de expressões regulares: um analisador Wiki” na página 92)
- WordSearch (“Exemplo de entrada de mouse: WordSearch” na página 595)

Exemplos práticos também são encontrados em muitos artigos do Quick Start no Flash Developer Center e no Flex Developer Center.

Execução de exemplos do ActionScript 3.0 no Flash Professional

Use um dos seguintes procedimentos (dependendo do tipo de exemplo) para executá-lo utilizando o Flash Professional.

Executando um exemplo de snippet de código no Flash Professional

Para executar um exemplo de snippet de código no Flash Professional:

1. Selecione Arquivo > Novo.
2. Na caixa de diálogo Novo documento, selecione Documento Flash e clique em OK.

   Uma nova janela do Flash será exibida.
3. Clique no primeiro quadro da primeira camada do painel Linha de tempo.
4. No painel Ações, digite ou cole o exemplo de snippet de código.
5. Selecione Arquivo > Salvar. Atribua um nome ao arquivo e clique em OK.
6. Para testar o exemplo, selecione Controlar > Testar filme.

Execução de um exemplo com base em classe no Flash Professional

Para executar um exemplo com base em classe no Flash Professional:

1. Selecione Arquivo > Novo.
2. Na caixa de diálogo Novo documento, selecione Arquivo ActionScript e clique em OK. Uma nova janela do editor será exibida.
3. Copie o código de exemplo com base em classe e cole-o na janela do editor.

   Se a classe for a classe principal document do programa, ela deverá estender a classe MovieClip:

```
import flash.display.MovieClip;
public class Example1 extends MovieClip{
//...
}
```

Certifique-se também de que todas as classes a que o exemplo faz referência sejam declaradas com instruções `import`.

4 Selecione Arquivo > Salvar. Dê ao arquivo o mesmo nome que a classe no exemplo (por exemplo, ContextMenuExample.as).

Nota: *Alguns dos exemplos com base em classe, como o exemplo de classe flashx.textLayout.container.ContainerController, incluem diversos níveis na declaração do pacote (`package flashx.textLayout.container.examples {`). Para esses exemplos, salve os arquivos em um subdiretório correspondente à declaração do pacote (flashx/textLayout/container/examples), ou remova o nome do pacote (de forma que o ActionScript inicie com `package {` somente) para que você possa testar o arquivo a partir de qualquer local.*

5 Selecione Arquivo > Novo.

6 Na caixa de diálogo Novo documento, selecione Documento Flash (ActionScript 3.0) e clique em OK. Uma nova janela do Flash será exibida.

7 No painel de Propriedades, campo Classe do documento, insira o nome da classe de exemplo, que deve corresponder ao nome do arquivo de origem do ActionScript que você acabou de salvar (por exemplo, ContextMenuExample).

8 Selecione Arquivo > Salvar. Dê ao arquivo FLA o mesmo nome que a classe no exemplo (por exemplo, ContextMenuExample.fla).

9 Para testar o exemplo, selecione Controlar > Testar filme.

Execução de um exemplo prático no Flash Professional

Normalmente, os exemplos práticos são distribuídos como arquivos ZIP. Para executar um exemplo prático usando o Flash Professional:

1 Descompacte o arquivo ZIP em outra pasta de sua escolha.

2 No Flash Professional, selecione Arquivo > Abrir.

3 Navegue até a pasta em que você descompactou o arquivo. Selecione o arquivo FLA nessa pasta e clique em Abrir.

4 Para testar o exemplo, selecione Controlar > Testar filme.

Execução de exemplos do ActionScript 3.0 no Flash Builder

Use um dos seguintes procedimentos (dependendo do tipo de exemplo) para executá-lo utilizando o Flash Builder.

Execução de um exemplo de snippet de código no Flash Builder

Para executar um exemplo de snippet de código no Flash Builder:

1 Crie um novo Projeto Flex (selecione Arquivo > Novo > Projeto Flex), ou, dentro de um projeto Flex existente, crie um novo Aplicativo MXML (selecione Arquivo > Novo > Aplicativo MXML). Dê ao projeto ou aplicativo um nome descritivo (como ContextMenuExample).

2 Dentro do arquivo MXML gerado, adicione uma marca `<mx:Script>`.

3 Cole o conteúdo do exemplo do snippet de código entre as marcas `<mx:Script>` e `</mx:Script>`. Salve o arquivo MXML.

4 Para executar o exemplo, selecione a opção de menu Executar > Executar para o arquivo MXML principal (como Executar > Executar ContextMenuExample).

Execução de um exemplo com base em classe no Flash Builder

Para executar um exemplo com base em classe no Flash Builder:

1 Selecione Arquivo > Novo > Projeto ActionScript.

2 Insira o nome da classe primária (como ContextMenuExample) no campo Nome do Projeto. Use os valores padrão nos outros campos (ou altere-os de acordo com o seu ambiente específico). Clique em Concluir para criar o projeto e o arquivo principal do ActionScript.

3 Apague todo o conteúdo gerado do arquivo ActionScript. Cole o código de exemplo, incluindo os comandos de pacote e importação, no arquivo ActionScript e salve o arquivo.

Nota: *Alguns dos exemplos com base em classe, como o exemplo de classe flashx.textLayout.container.ContainerController, incluem diversos níveis na declaração do pacote (`package flashx.textLayout.container.examples {`). Para esses exemplos, salve os arquivos em um subdiretório correspondente à declaração do pacote (flashx/textLayout/container/examples), ou remova o nome do pacote (de forma que o ActionScript inicie com `package {` somente) para que você possa testar o arquivo a partir de qualquer local.*

4 Para executar o exemplo, selecione a opção de menu Executar > Executar para o nome da classe do ActionScript (como Executar > Executar ContextMenuExample).

Execução de um exemplo prático no Flash Builder

Normalmente, os exemplos práticos são distribuídos como arquivos ZIP. Para executar um exemplo prático usando o Flash Builder:

1 Descompacte o arquivo ZIP em outra pasta de sua escolha. Dê à pasta um nome descritivo (como ContextMenuExample).

2 No Flash Builder, selecione Arquivo > Novo Projeto Flex. Na seção Local do Projeto, clique em Localizar e selecione a pasta que contém os arquivos de exemplo. No campo Nome do Projeto, insira o nome da pasta (como ContextMenuExample). Use os valores padrão nos outros campos (ou altere-os de acordo com o seu ambiente específico). Clique em Avançar para continuar.

3 No painel Saída, clique em Avançar para aceitar o valor padrão.

4 No painel Caminhos de Origem, clique no botão Localizar ao lado do campo Arquivo Principal do Aplicativo. Selecione o arquivo MXML principal de exemplo na pasta de exemplo. Clique em Concluir para criar os arquivos do projeto.

5 Para executar o exemplo, selecione a opção de menu Executar > Executar para o arquivo MXML principal (como Executar > Executar ContextMenuExample).

Execução de exemplos do ActionScript 3.0 em dispositivos móveis

Você pode executar os exemplos de código do ActionScript 3.0 em dispositivos móveis que reconhecem o Flash Player 10.1. Entretanto, você normalmente executa um exemplo de código para aprender como funcionam determinadas classes e determinados métodos. Nesse caso, execute o exemplo em um dispositivo que não seja móvel, como um computador de mesa. No computador de mesa, você pode usar comandos de rastreamento e outras ferramentas de depuração no Flash Professional ou no Flash Builder para compreender melhor o exemplo de código.

Se quiser executar o exemplo em um dispositivo móvel, você pode copiar os arquivos no dispositivo ou em um servidor Web. Para copiar arquivos no dispositivo e executar o exemplo no navegador, faça o seguinte:

1. Crie o arquivo SWF seguindo as instruções em "Execução de exemplos do ActionScript 3.0 no Flash Professional" na página 1141 ou em "Execução de exemplos do ActionScript 3.0 no Flash Builder" na página 1142. No Flash Professional, crie o arquivo SWF quando selecionar Controlar > Testar Filme. No Flash Builder, crie o arquivo SWF quando executar, depurar ou criar o seu projeto do Flash Builder.
2. Copie o arquivo SWF em um diretório no dispositivo móvel. Use o software que acompanha o dispositivo para copiar o arquivo.
3. Na barra de endereços do navegador no dispositivo móvel, digite o arquivo:// URL para o arquivo SWF. Por exemplo, digite `file:://applications/myExample.swf`.

Para copiar arquivos em um servidor Web e executar o exemplo no navegador do dispositivo, faça o seguinte:

1. Crie um arquivo SWF e um arquivo HTML. Primeiro, siga as instruções em "Execução de exemplos do ActionScript 3.0 no Flash Professional" na página 1141 ou em "Execução de exemplos do ActionScript 3.0 no Flash Builder" na página 1142. No Flash Professional, a seleção de Controlar > Testar Filme cria apenas o arquivo SWF. Para criar ambos os arquivos, primeiro selecione Flash e HTML na guia Formatos da caixa de diálogo Publicar Configurações. Em seguida, selecione Arquivo > Publicar para criar os arquivos HTML e SWF. No Flash Builder, crie os arquivos SWF e HTML quando executar, depurar ou criar o seu projeto do Flash Builder.
2. Copie os arquivos SWF e HTML em um diretório do servidor Web.
3. Na barra de endereços do navegador no dispositivo móvel, digite o endereço HTTP do arquivo SWF. Por exemplo, digite `http://www.myWebServer/examples/myExample.html`.

Antes de executar um exemplo em um dispositivo móvel, considere cada um dos seguintes aspectos.

Tamanho do palco

O tamanho do palco usado na execução de um exemplo em um dispositivo móvel é muito menor do que quando você utiliza um dispositivo que não seja móvel. Muitos exemplos não exigem um tamanho específico de palco. Ao criar o arquivo SWF, especifique um tamanho de palco adequado ao seu dispositivo. Por exemplo, especifique 176 x 208 pixels.

A finalidade dos exemplos práticos no Guia de desenvolvimento do ActionScript 3.0 é ilustrar diferentes conceitos e classes do ActionScript 3.0. Suas interfaces de usuário são criadas para ter uma boa aparência e funcionar bem em um computador de mesa ou em um laptop. Embora os exemplos funcionem em dispositivos móveis, o tamanho do palco e o design da interface de usuário não são adequados à tela pequena. A Adobe recomenda que você execute os exemplos práticos em um computador para conhecer o ActionScript e, depois, use os snippets de código pertinentes nos seus aplicativos móveis.

Campos de texto em vez de comandos de rastreamento.

Ao executar um exemplo em um dispositivo móvel, não é possível ver o resultado dos comandos de rastreamento do exemplo. Para ver o resultado, crie uma instância da classe TextField. Depois, anexe o texto dos comandos de rastreamento à propriedade `text` do campo de texto.

Você pode usar a seguinte função para criar um campo de texto para usar no rastreamento:

```
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;
}
```

Por exemplo, acrescente esta função à classe de documentos como uma função privada. Em seguida, em outros métodos da classe de documentos, rastreie os dados com códigos como os seguintes:

```
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("");
```

O método `appendText()` aceita apenas um valor como parâmetro. Esse valor é uma sequência de caracteres (uma instância de String ou um literal de sequência de caracteres). Para imprimir o valor de uma variável que não seja uma sequência de caracteres, primeiro é necessário converter o valor em uma String. A maneira mais fácil de fazer isso é chamar o método `toString()` do objeto:

```
var albumYear:int = 1999;
traceField.appendText("albumYear = ");
traceField.appendText(albumYear.toString());
```

Tamanho do texto

Muitos exemplos usam campos de texto para ajudar a ilustrar um conceito. Às vezes, o ajuste do tamanho do texto no campo de texto melhora a legibilidade em um dispositivo móvel. Por exemplo, se um exemplo usar uma instância de campo de texto chamado `myTextField`, altere o tamanho do texto com o seguinte código:

```
// Use a text size that works well on the device.
var myFormat:TextFormat = new TextFormat();
myFormat.size = 18;
myTextField.defaultTextFormat = myFormat
```

Captura da entrada do usuário

O sistema operacional e o navegador móveis capturam alguns eventos de entrada do cliente que não são recebidos pelo conteúdo SWF. O comportamento específico depende do sistema operacional e do navegador, mas pode causar comportamentos inesperados quando você executar os exemplos em um dispositivo móvel. Para obter mais informações, consulte "Precedência de KeyboardEvent" na página 579.

Além disso, as interfaces de usuário de muitos exemplos foram criadas para computadores de mesa ou laptops. Por exemplo, a maioria dos exemplos práticos no Guia do desenvolvedor do ActionScript 3.0 é adequada para visualização em computadores de mesa. Portanto, às vezes, nem todo o palco pode ser visto na tela do dispositivo móvel. A capacidade de deslocamento horizontal pelo conteúdo do navegador depende do navegador. Além disso, os exemplos não foram criados para capturar e tratar eventos de rolagem ou deslocamento horizontal. Portanto, as interfaces de usuário de alguns exemplos não são adequadas para execução em telas pequenas. A Adobe recomenda que você execute os exemplos em um computador para conhecer o ActionScript e, depois, use os snippets de código pertinentes nos seus aplicativos móveis.

Para obter mais informações, consulte "Visão panorâmica e rolagem de objetos de exibição" na página 181.

Tratamento do foco

Alguns exemplos exigem que você coloque o foco em um campo. Levando o foco a um campo, você pode, por exemplo, inserir textos ou selecionar um botão. Para levar o foco a um campo, use o ponteiro do dispositivo móvel, como um utensílio de escrita ou o dedo. Você também pode usar as teclas de navegação do dispositivo móvel para levar o foco a um campo. Para selecionar um botão no qual o foco está, use a tecla Selecionar do dispositivo móvel da mesma forma que usaria o Enter em um computador. Em alguns dispositivos, você pode selecionar um botão batendo duas vezes nele.

Para obter mais informações sobre foco, consulte "Gerenciamento do foco" na página 574.

Tratamento de eventos do mouse

Muitos exemplos recebem eventos do mouse. Em um computador, esses eventos de mouse podem ocorrer, por exemplo, quando um usuário passa com o cursor sobre um objeto de exibição ou quando ele clica com o botão do mouse sobre esse objeto. Nos dispositivos móveis, os eventos gerados pelo uso de ponteiros (como um utensílio de escrita ou o dedo), são chamados 'eventos de toque'. O Flash Player 10.1 correlaciona os eventos de toque aos eventos de mouse. Essa correlação garante que o conteúdo SWF desenvolvido antes do Flash Player 10.1 continue funcionando. Portanto, os exemplos funcionam quando você utiliza um ponteiro ou arrasta um objeto de exibição.

Desempenho

Os dispositivos móveis têm menos capacidade de processamento que os computadores de mesa. Possivelmente, a velocidade de alguns exemplos que consomem muita CPU será menor em dispositivos móveis. Por exemplo, o exemplo em "Exemplo de API de desenho: gerador visual algorítmico" na página 237 faz muitos cálculos e desenhos na entrada em cada quadro. A execução desse exemplo em um computador ilustra várias APIs de desenho. Entretanto, o exemplo não é adequado em alguns dispositivos móveis devido às suas limitações de velocidade.

Para obter mais informações sobre o desempenho em dispositivos móveis, consulte *Otimização do desempenho para a plataforma Flash*.

Melhores práticas

Os exemplos não levam em conta as melhores práticas de desenvolvimento de aplicativos para dispositivos móveis. As limitações de memória e capacidade de processamento dos dispositivos móveis exigem uma atenção especial. Da mesma forma, a interface de usuário para telas pequenas tem necessidades diferentes em relação à tela de um computador de mesa. Para obter mais informações sobre o desenvolvimento de aplicativos para dispositivos móveis, *consulte Otimização do desempenho para a plataforma Flash*.

Capítulo 66: Suporte SQL em bancos de dados locais

O Adobe AIR inclui um mecanismo de banco de dados SQL com suporte a bancos de dados locais SQL com diversos recursos padrão SQL, usando o sistema de banco de dados de código livre SQLite. O tempo de execução não especifica como ou onde os dados do banco de dados estão armazenados no sistema de arquivos. Cada banco de dados é armazenado totalmente em um único arquivo. Um desenvolvedor pode especificar o local no sistema de arquivos em que o arquivo do banco de dados é armazenado e um aplicativo do AIR único pode acessar um ou mais bancos de dados separados (ou seja, arquivos de banco de dados separados). Este documento descreve a sintaxe SQL e o suporte a tipo de dados para bancos de dados SQL locais do Adobe AIR. Este documento não deve servir como uma ampla referência a SQL. Na verdade, ele descreve detalhes específicos do dialeto SQL aos quais o Adobe AIR oferece suporte. O tempo de execução oferece suporte à maior parte do dialeto SQL padrão SQL-92. Como há muitas referências, sites, livros e materiais de treinamento para aprender SQL, este documento não deve ser uma ampla referência ou tutorial sobre SQL. Na verdade, este documento se concentra especialmente na sintaxe SQL para a qual o Apollo AIR oferece suporte, além das diferenças entre o SQL-92 e o dialeto SQL para o qual há suporte.

Convenções de definição de instruções SQL

Nas definições de instrução deste documento, são usadas as seguintes convenções:

- Uso de maiúsculas e minúsculas
  - UPPER CASE – palavras-chave SQL literais são escritas todas em maiúsculas.
  - lower case – termos de alocador de espaço ou nomes de cláusula são escritos todos em minúsculas.
- Caracteres de definição
  - = – indica uma definição de cláusula ou instrução.
- Caracteres de agrupamento e alternativos
  - – o caractere de pipe é usado entre opções e pode ser lido como "ou"
  - [] -– itens entre colchetes são opcionais; eles podem conter um único item ou um conjunto de itens alternativos.
  - () – parênteses entre um conjunto de alternativas (um conjunto de itens separados por caracteres de pipe) designam um grupo obrigatório de itens, ou seja, um conjunto de itens que são valores possível para um único item obrigatório.
- Quantificadores
  - + - um caractere de soma seguido de um item entre parênteses indica que o item anterior pode ocorrer uma ou mais vezes.
  - * - um caractere de asterisco seguido de um item entre colchetes indica que o item anterior (entre colchetes) pode ocorrer 0 ou mais vezes.
- Caracteres literais
  - * – um caractere de asterisco usado em um nome de coluna ou entre parênteses seguido de um nome de função significa um asterisco literal e não o quantificador "0 ou mais".
  - . . - Um caractere de ponto final representa um ponto final literal.
  - , – um caractere de vírgula representa uma vírgula literal.

- () – um par de parênteses entre uma única cláusula ou item indica que os parênteses são caracteres de parênteses literais, obrigatórios.
- Outros caracteres – exceto quando indicado o contrário, os demais caracteres representam esses caracteres literais.

Sintaxe SQL para a qual há suporte

As seguintes litas de sintaxe SQL são suportadas pelo mecanismo de banco de dados SQL Adobe AIR. Essas listagens estão divididas em explicações de tipos de instrução e cláusula diferentes, expressões, funções internas e operadores. São abordados os tópicos a seguir:

- Sintaxe SQL geral
- Instruções para manipulação de dados (SELECT, INSERT, UPDATE e DELETE)
- Instruções para definição de dados (instruções de CREATE, ALTER e DROP para tabelas, índices, exibições e disparadores)
- Instruções especiais e cláusulas
- Funções internas (agregada, escalar e funções de formatação de data/hora)
- Operadores
- Parâmetros
- Recursos SQL para os quais não há suporte
- Recursos SQL adicionais

Sintaxe SQL geral

Além da sintaxe específica de várias instruções e expressões, estão são as regras gerais da sintaxe SQL:

Diferenciação entre maiúsculas e minúsculas Instruções SQL, inclusive nomes de objeto, não diferenciam maiúsculas e minúsculas. No entanto, as instruções SQL costumam ser escritas com palavras-chave SQL em maiúsculas, e este documento usa essa convenção. Embora a sintaxe SQL não diferencie maiúsculas de minúsculas, os valores de texto literal em SQL as diferenciam, e as operações de classificação podem diferenciá-las, conforme especificado pela sequência de intercalação definida para uma coluna ou operação. Para obter mais informações, consulte COLLATE.

Espaço em branco Um caractere de espaço em branco (como, por exemplo, espaço, tabulação, nova linha etc.) deve ser usado para separar palavras individuais em uma instrução SQL. No entanto, o espaço em branco é opcional entre palavras e símbolos. O tipo e a quantidade de caracteres espaço em branco em uma instrução SQL não são significativos. É possível usar espaço em branco como, por exemplo, recuos e quebras de linha para formatar as instruções SQL, tendo em vista maior legibilidade sem afetar o significado da instrução.

Instruções para manipulação de dados

As instruções para manipulação de dados são as instruções SQL mais usadas. Elas são usadas para recuperar, adicionar, modificar e remover dados das tabelas de banco de dados. Estas são as instruções para manipulação de dados para as quais há suporte: SELECT, INSERT, UPDATE e DELETE.

SELECT

A instrução SELECT é usada para consultar o banco de dados. O resultado de uma instrução SELECT é zero ou mais linhas de dados em que cada uma tem um número de colunas fixo. O número de colunas no resultado é especificado pelo nome da coluna result ou pela lista de expressão entre as palavras-chave SELECT e as opcionais FROM.

```
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
```

Qualquer expressão arbitrária pode ser usada como resultado. Caso uma expressão de resultado seja *, todas as colunas de todas as tabelas serão substituídas por essa expressão. Caso a expressão seja o nome de uma tabela seguido de .*, o resultado será igual a todas as colunas nessa única tabela.

A palavra-chave DISTINCT faz com que um subconjunto de linhas de resultado seja retornado, no qual cada linha de resultado é diferente. Valores NULL não são tratados como distintos entre si. O comportamento padrão é de que todas as linhas de resultado sejam retornadas, as quais podem ser explicitadas com a palavra-chave ALL.

A consulta é executada em uma ou mais tabelas especificadas após a palavra-chave FROM. Caso vários nomes de tabela estejam separados por vírgulas, a consulta usa a junção cruzada das várias tabelas. A sintaxe JOIN também pode ser usada para especificar como as tabelas são unidas. O único tipo de junção externa para o qual há suporte é LEFT OUTER JOIN. A expressão da cláusula ON em join-args deve ser resolvida como um valor booliano. Uma subconsulta entre parênteses pode ser usada como uma tabela na cláusula FROM. Toda a cláusula FROM pode ser omitida, no caso de o resultado ser uma única linha com os valores da lista de expressões result.

A cláusula WHERE é usada para limitar o número de linhas recuperadas pela consulta. As expressões de cláusula WHERE devem ser resolvidas como um valor booliano. Como a filtragem de cláusula WHERE é realizada antes de qualquer agrupamento, as expressões de cláusula WHERE talvez não incluam funções agregadas.

A cláusula GROUP BY faz com que uma ou mais linhas do resultado sejam combinadas com uma única linha de saída. Uma cláusula GROUP BY é especialmente útil quando o resultado contém funções agregadas. As expressões na cláusula GROUP BY não precisam ser expressões exibidas na lista de expressões SELECT.

A cláusula HAVING é semelhante a WHERE ao limitar as linhas retornadas pela instrução. No entanto, a cláusula HAVING se aplica após a ocorrência de qualquer agrupamento especificado por uma cláusula GROUP BY. Consequentemente, a expressão HAVING pode se referir a valores que incluem funções agregadas. Uma expressão de cláusula HAVING não precisa ser exibida na lista SELECT. Assim como uma expressão WHERE, uma expressão HAVING deve ser resolvida como um valor booliano.

A cláusula ORDER BY faz com que as linhas de saída sejam classificadas. O argumento sort-expr-list da cláusula ORDER BY é uma lista de expressões usadas como chave da classificação. As expressões não precisam fazer parte do resultado de uma simples SELECT, mas em uma SELECT composta (uma SELECT usando um dos operadores compound-op), cada expressão de classificação deve corresponder exatamente a uma das colunas de resultado. Cada expressão de classificação pode ser seguida de uma cláusula sort-order que consiste na palavra-chave COLLATE e no nome de uma função de intercalação usada para classificar o texto e/ou a palavra-chave ASC ou DESC para especificar a ordem de classificação (crescente ou decrescente). A sort-order pode ser omitida e o padrão (ordem crescente) é usado. Para obter uma definição da cláusula COLLATE e das funções de intercalação, consulte COLLATE. A sort-order pode ser omitida e o padrão (ordem crescente) é usado. Para obter uma definição da cláusula COLLATE e das funções de intercalação, consulte COLLATE.

A cláusula LIMIT coloca um limite superior sobre o número de linhas retornadas no resultado. Uma LIMIT negativa não indica nenhum limite superior. A OFFSET opcional que segue LIMIT especifica quantas linhas ignorar no início do conjunto de resultados. Em uma consulta SELECT composta, a cláusula LIMIT talvez só seja exibida após a instrução SELECT final, e o limite é aplicado a toda a consulta. Observe que, caso a palavra-chave OFFSET seja usada na cláusula LIMIT, o limite é o primeiro inteiro e a distância é o segundo inteiro. Caso seja usada uma vírgula no lugar da palavra-chave OFFSET, a distância é o primeiro número e o limite, o segundo. Essa contradição aparente é intencional — ela maximiza a compatibilidade com sistemas de banco de dados SQL herdados.

Uma SELECT composta é formada por duas ou mais instruções SELECT simples conectadas por um dos operadores UNION, UNION ALL, INTERSECT ou EXCEPT. Em uma SELECT composta, todas as instruções SELECT constituintes devem especificar o mesmo número de colunas de resultado. Só pode haver uma única cláusula ORDER BY após a instrução SELECT final (e antes da única cláusula LIMIT, caso haja uma especificada). Os operadores UNION e UNION ALL integram os resultados das instruções SELECT anterior e posterior em uma única tabela. A diferença é que na UNION, todas as linhas de resultado são distintas, mas em UNION ALL talvez haja cópias. O operador INTERSECT usa a intersecção dos resultados das instruções SELECT anterior e posterior. EXCEPT usa o resultado da SELECT anterior após a remoção dos resultados da SELECT seguinte. Quando três ou mais instruções SELECT estão conectadas a uma composta, elas são agrupadas da primeira à última.

Para obter uma definição das expressões permitidas, consulte Expressões

A partir do AIR 2.5, o operador SQL CAST conta com suporte durante a leitura para converter dados BLOB em objetos ByteArray do ActionScript. Por exemplo, o código a seguir lê dados brutos que não estão armazenados no formato AMF e os armazena em um objeto ByteArray:

```
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

A instrução INSERT acompanha duas formas básicas, sendo usada para preencher tabelas com dados.

```
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
```

A primeira forma (com a palavra-chave VALUES) cria uma única linha nova em uma tabela existente. Caso nenhuma column-list seja especificada, o número de valores deve ser igual ao número de colunas na tabela. Caso uma column-list seja especificada, o número de valores deve corresponder ao número de colunas especificadas. As colunas da tabela que não são exibidas na lista são preenchidas com o valor definido padrão quando a tabela é criada, ou com NULL caso nenhum valor padrão seja definido.

A segunda forma da instrução INSERT usa os dados de uma instrução SELECT. O número de colunas no resultado da SELECT deve corresponder exatamente ao número de colunas na tabela caso column-list não esteja especificada, ou deve corresponder ao número de colunas nomeadas na column-list. É criada uma nova entrada na tabela para todas as linhas do resultado de SELECT. A SELECT pode ser simples ou composta. Para obter uma definição das instruções SELECT permitidas, consulte SELECT.

A conflict-algorithm opcional permite a especificação de um algoritmo para resolução de conflitos de restrição a ser usado durante esse comando único. Para obter uma explicação e definição do conflito de algoritmos, consulte "Instruções especiais e cláusulas" na página 1159.

As duas formas REPLACE INTO da instrução são equivalentes ao uso da forma INSERT [OR conflict-algorithm] padrão com o algoritmo de conflito REPLACE (ou seja, a forma INSERT OR REPLACE...).

As duas formas REPLACE INTO da instrução são equivalentes ao uso da forma INSERT [OR conflict-algorithm] padrão com o algoritmo de conflito REPLACE (ou seja, a forma INSERT OR REPLACE...).

UPDATE

O comando update altera os registros existentes em uma tabela.

```
sql-statement  ::=  UPDATE [database-name.] table-name SET column1=value1, column2=value2,...
[WHERE expr]
```

O comando consiste na palavra-chave UPDATE seguida do nome do tabela cujos registros você deseja atualizar. Depois da palavra-chave SET, forneça o nome da coluna e o valor para o qual a coluna deve ser alterada, como uma lista separada por vírgulas. A expressão da cláusula WHERE indica a linha ou linhas em que os registros são atualizados.

DELETE

O comando de exclusão é usado para remover registros de uma tabela.

```
sql-statement  ::=  DELETE FROM [database-name.] table-name [WHERE expr]
```

O comando consiste nas palavras-chave DELETE FROM seguidas do nome da tabela da qual os registros devem ser removidos.

Sem uma cláusula WHERE, todas as linhas da tabela são removidas. Caso uma cláusula WHERE seja fornecida, apenas essas linhas correspondentes à expressão são removidas. A expressão da cláusula WHERE deve ser resolvida como um valor booliano. Para obter uma definição das expressões permitidas, consulte Expressões

Instruções para definição de dados

As instruções para definição dos dados são usadas para criar, modificar e remover objetos de banco de dados como tabelas, exibições, índices e disparadores. Estas são as instruções para definição de dados para as quais há suporte:

- Tabelas:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE

- Índices:
  - CREATE INDEX
  - DROP INDEX
- Exibições:
  - CREATE VIEWS
  - DROP VIEWS
- Disparadores:
  - CREATE TRIGGERS
  - DROP TRIGGERS

CREATE TABLE

Uma instrução CREATE TABLE consiste nas palavras-chave CREATE TABLE seguidas do nome da nova tabela e, em seguida, de uma lista das definições e restrições da coluna (entre parênteses). O nome da tabela pode ser um identificador ou uma string.

```
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]*
```

Cada definição de coluna é o nome da coluna seguido do tipo de dados da coluna e, em seguida, de uma ou mais restrições de coluna opcionais. O tipo de dados da coluna restringe quais dados podem ser armazenados nessa coluna. Se for feita uma tentativa de armazenar um valor em uma coluna com um tipo de dados diferente, o tempo de execução converterá o valor no tipo apropriado, se possível, ou apresentará um erro. Consulte Suporte ao tipo de dados para obter detalhes.

A restrição de coluna NOT NULL indica que a coluna não pode conter valores NULL.

Uma restrição UNIQUE faz com que seja criado um índice na(s) coluna(s) especificada(s). Esse índice deve conter chaves exclusivas — duas linhas não podem conter valores duplicados ou combinações de valores para a(s) coluna(s) especificada(s). Uma instrução CREATE TABLE pode ter várias restrições UNIQUE, inclusive várias colunas com uma restrição UNIQUE na definição da coluna e/ou várias restrições UNIQUE no nível da tabela.

Uma restrição CHECK define uma expressão avaliada e que deve ser verdadeira para que os dados de uma linha sejam inseridos ou atualizados. A expressão CHECK deve ser resolvida como um valor booliano.

Uma cláusula COLLATE numa definição de coluna especifica qual função de colação de texto deverá ser usado ao comparar entradas de texto na coluna. Por padrão, a função de intercalação BINARY é usada. Para obter detalhes sobre a cláusula COLLATE e as funções de intercalação, consulte COLLATE.

A restrição DEFAULT especifica um valor padrão a ser usado com INSERT. O valor pode ser NULL, uma constante de string ou um número. O valor padrão também pode ser uma das palavras-chave especiais que não diferenciam maiúsculas de minúsculas CURRENT_TIME, CURRENT_DATE ou CURRENT_TIMESTAMP. Caso o valor seja NULL, uma constante de string, ou um número, ele é literalmente inserido na coluna sempre que uma instrução INSERT não especificar um valor da coluna. Caso o valor seja CURRENT_TIME, CURRENT_DATE ou CURRENT_TIMESTAMP, a data e/ou hora UTC atual é inserida na coluna. Em CURRENT_TIME, o formato é HH:MM:SS. Em CURRENT_DATE, o formato é YYYY-MM-DD. O formato de CURRENT_TIMESTAMP é YYYY-MM-DD HH:MM:SS.

Especificar uma PRIMARY KEY normalmente apenas cria um índice UNIQUE na(s) coluna(s) correspondente(s). No entanto, caso a restrição PRIMARY KEY esteja em uma única coluna cujo tipo de dados seja INTEGER (ou um de seus sinônimos, tais como int), essa coluna será usada pelo banco de dados como a chave primária real da tabela. Isso significa que a coluna só pode manter valores de inteiro exclusivos. (Observe que, em muitas implementações do SQLite, apenas o tipo de coluna INTEGER faz com que a coluna sirva como chave primária interna, mas, no Adobe AIR, os sinônimos de INTEGER, como int, também especificam esse comportamento).

Se uma tabela não tiver uma coluna INTEGER PRIMARY KEY, uma chave de inteiro será gerada automaticamente quando uma linha for inserida. A chave primária de uma linha pode ser sempre acessada usando um dos nomes especiais ROWID, OID ou _ROWID_. Esses nomes podem ser usados independentemente da declaração explícita de INTEGER PRIMARY KEY ou de um valor gerado internamente. Entretanto, caso a tabela tenha uma INTEGER PRIMARY KEY explícita, o nome da coluna nos dados de resultados é o nome real da coluna e não o nome especial.

Uma coluna INTEGER PRIMARY KEY também pode incluir a palavra-chave AUTOINCREMENT. Quando a palavra-chave AUTOINCREMENT for usada, o banco de dados gera automaticamente e insere uma chave inteira incrementada sequencialmente na coluna INTEGER PRIMARY KEY quando executa uma instrução INSERT que não especifica um valor explícito para a coluna.

Só pode haver uma restrição PRIMARY KEY em uma instrução CREATE TABLE. Ela pode fazer parte da definição de uma coluna ou de uma restrição PRIMARY KEY única no nível da tabela. Implicitamente, uma coluna de chave primária é NOT NULL.

A conflict-clause opcional que segue muitas restrições permite a especificação de um algoritmo para resolução de conflitos de restrição padrão alternativo para essa restrição. O padrão é ABORT. restrições diferentes dentro da mesma tabela podem ter algoritmos para resolução de conflitos padrão diferentes. Caso uma instrução INSERT ou UPDATE especifique um algoritmo para resolução de conflitos diferente, esse algoritmo é usado em lugar do especificado na instrução CREATE TABLE. Consulte a seção ON CONFLICT de "Instruções especiais e cláusulas" na página 1159 para obter informações adicionais.

restrições adicionais como, por exemplo, FOREIGN KEY não resultam em erro, mas o tempo de execução as ignora.

Caso ocorra palavra-chave TEMP ou TEMPORARY entre CREATE e TABLE, a tabela criada permanece visível apenas dentro da mesma conexão de banco de dados (ocorrência de SQLConnection). Ela é excluída automaticamente quando a conexão de banco de dados é fechada. Todos os índices criados em uma tabela temporária também são temporários. As tabelas e os índices temporários são armazenados em um arquivo separado distinto do arquivo de banco de dados principal.

Caso o prefixo opcional database-name seja especificado, a tabela é criada em um banco de dados nomeado (um banco de dados conectado à instância de SQLConnection pela chamada do método attach() com o nome do banco de dados especificado). É um erro especificar um prefixo database-name e a palavra-chave TEMP, a menos que o prefixo database-name seja temp. Caso nenhum nome de banco de dados seja especificado e a palavra-chave TEMP não esteja presente, a tabela é criada no banco de dados principal (o banco de dados conectado à instância de SQLConnection usando o método open() ou openAsync()).

não há nenhum limite arbitrário quanto ao número de colunas ou ao número de restrições em uma tabela. Também não existe nenhum limite arbitrário quanto à quantidade de dados em uma linha.

A forma CREATE TABLE AS define a tabela como o conjunto de resultados de uma consulta. Os nomes das colunas da tabela são os nomes das colunas no resultado.

Caso a cláusula opcional IF NOT EXISTS esteja presente e já haja outra tabela com o mesmo nome, o banco de dados ignora o comando CREATE TABLE.

Uma tabela pode ser removida usando a instrução DROP TABLE, e alterações limitadas podem ser feitas usando a instrução ALTER TABLE.

ALTER TABLE

O comando ALTER TABLE permite ao usuário renomear ou adicionar uma nova coluna a uma tabela existente. Não é possível remover uma coluna de uma tabela.

```
sql-statement ::= ALTER TABLE [database-name.] table-name alteration
alteration    ::= RENAME TO new-table-name
alteration    ::= ADD [COLUMN] column-def
```

A sintaxe RENAME TO é usada para renomear a tabela identificada por [database-name.] table-name para new-table-name. Esse comando não pode ser usado para mover uma tabela entre bancos de dados anexados, e sim apenas para renomear uma tabela dentro do mesmo banco de dados.

Caso a tabela que está sendo renomeada tenha disparadores ou índices, eles permanecem anexados à tabela depois que ela é renomeada. No entanto, caso haja alguma definição ou instrução de exibição executada por disparadores referente à tabela que está sendo renomeada, eles não serão modificados automaticamente para usar o novo nome da tabela. Caso uma tabela renomeada tenha exibições ou disparadores, você deve ignorar e recriar manualmente os disparadores ou as definições de exibição usando o novo nome da tabela.

A sintaxe ADD [COLUMN] é usada para adicionar uma nova coluna a uma tabela existente. A nova coluna sempre é acrescentada ao final da lista de colunas existentes. A cláusula column-def pode assumir qualquer uma das formas permitidas em uma instrução CREATE TABLE, com as seguintes restrições:

- A coluna talvez não tenha uma restrição PRIMARY KEY ou UNIQUE.
- A coluna talvez não tenha um valor padrão igual a CURRENT_TIME, CURRENT_DATE ou CURRENT_TIMESTAMP.
- Caso seja especificada uma restrição NOT NULL, a coluna deve ter um valor padrão diferente de NULL.

O momento da execução da instrução ALTER TABLE não afetado pela quantidade de dados na tabela.

DROP TABLE

A instrução DROP TABLE remove uma tabela adicionada com uma instrução CREATE TABLE. A tabela com a table-name especificada é a tabela ignorada. Ela é totalmente removida do banco de dados e do arquivo em disco. Ela não pode ser recuperada. Todos os índices associados à tabela também são excluídos.

```
sql-statement  ::=  DROP TABLE [IF EXISTS] [database-name.] table-name
```

Por padrão, a instrução DROP TABLE não reduz o tamanho do arquivo de banco de dados. O espaço livre no banco de dados é mantido e usado em operações INSERT subsequentes. Para remover o espaço livre no banco de dados, use o método SQLConnection.clean(). Caso o parâmetro autoClean esteja definido como true quando o banco de dados for criado inicialmente, o espaço é automaticamente liberado.

A cláusula IF EXISTS opcional suprime o erro que normalmente aconteceria se a tabela não existisse.

CREATE INDEX

O comando CREATE INDEX consiste nas palavras-chave CREATE INDEX seguidas do nome do novo índice, a palavra-chave ON, o nome de uma tabela criada anteriormente a ser indexada e uma lista entre parênteses dos nomes das colunas na tabela cujos valores são usados na chave de índice.

```
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]
```

Cada nome de coluna pode ser seguido das palavras-chave ASC ou DESC para indicar a ordem de classificação, embora a designação da ordem de classificação seja ignorada pelo tempo de execução. A classificação é sempre feita na ordem crescente.

A cláusula COLLATE que segue cada nome de coluna define uma sequência de intercalação usada nos valores de texto da coluna. A sequência de intercalação padrão é a definida para a coluna na instrução CREATE TABLE. Caso nenhuma seja especificada, é usada a sequência de intercalação BINARY. Para obter uma definição da cláusula COLLATE e das funções de intercalação, consulte COLLATE.

Não há limites arbitrários quanto ao número de índices que podem ser anexados a uma única tabela. Também não existem limites para o número de colunas em um índice.

DROP INDEX

A instrução drop index remove um índice adicionado com a instrução CREATE INDEX. O índice especificado é removido completamente do arquivo de banco de dados. A única forma de recuperar o índice é digitar novamente o comando CREATE INDEX apropriado.

```
sql-statement ::= DROP INDEX [IF EXISTS] [database-name.] index-name
```

Por padrão, a instrução DROP INDEX não reduz o tamanho do arquivo de banco de dados. O espaço livre no banco de dados é mantido e usado em operações INSERT subsequentes. Para remover o espaço livre no banco de dados, use o método SQLConnection.clean(). Caso o parâmetro autoClean esteja definido como true quando o banco de dados for criado inicialmente, o espaço é automaticamente liberado.

CREATE VIEW

O comando CREATE VIEW atribui um nome a uma instrução SELECT predefinida. Esse novo nome pode ser usado em uma cláusula FROM de outra instrução SELECT em lugar do nome de uma tabela. As exibições costumam ser usadas para simplificar as consultas, integrando um conjunto complexo (e muito usado) de dados a uma estrutura que pode ser usada em outras operações.

```
sql-statement ::= CREATE [TEMP | TEMPORARY] VIEW [IF NOT EXISTS] [database-name.] view-name AS
select-statement
```

Caso ocorra a palavra-chave TEMP ou TEMPORARY entre CREATE e VIEW, a exibição criada só permanece visível para a ocorrência de SQLConnection que abriu o banco de dados, sendo excluída automaticamente quando ele é fechado.

Caso seja especificada uma [database-name], a exibição é criada no banco de dados nomeado (um banco de dados conectado à instância de SQLConnection usando o método attach(), com o argumento name especificado). É um erro especificar uma [database-name] e a palavra-chave TEMP, a menos que [database-name] seja temp. Caso nenhum nome de banco de dados seja especificado e a palavra-chave TEMP não esteja presente, a exibição é criada no banco de dados principal (o banco de dados conectado à instância de SQLConnection usando o método open() ou openAsync()).

As exibições são somente leitura. Uma instrução DELETE, INSERT ou UPDATE não pode ser usada em uma exibição, a menos que haja pelo menos um disparador do tipo associado (INSTEAD OF DELETE, INSTEAD OF INSERT, INSTEAD OF UPDATE) definido. Para informações sobre a criação de um gatilho para uma visualização, consulte CRIAR GATILHO.

Uma exibição é removida de um banco de dados usando a instrução DROP VIEW.

DROP VIEW

A instrução DROP VIEW remove uma exibição criada por uma instrução CREATE VIEW.

```
sql-statement ::= DROP VIEW [IF EXISTS] view-name
```

A view-name especificada é o nome da exibição a ser ignorada. Ela é removida do banco de dados, mas nenhum dado nas tabelas subjacentes é modificado.

CREATE TRIGGER

A instrução create trigger é usada para adicionar disparadores ao esquema do banco de dados. Um disparador é uma operação do banco de dados (a trigger-action) que é realizada automaticamente quando ocorre um evento de banco de dados especificado (o database-event).

```
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]*
```

O acionamento de um disparador é especificado sempre que ocorre uma DELETE, INSERT ou UPDATE em uma determinada tabela de banco de dados, ou sempre que uma UPDATE de uma ou mais colunas especificadas de uma tabela são atualizadas. Os disparadores são permanentes, a menos que a palavra-chave TEMP ou TEMPORARY seja usada. Nesse caso, o disparador é removido quando a conexão de banco de dados principal da instância de SQLConnection é fechada. Caso nenhum tempo seja especificado (BEFORE ou AFTER), o disparador assume o padrão BEFORE.

Como só há suporte para disparadores FOR EACH ROW, o texto FOR EACH ROW é opcional. Com um disparador FOR EACH ROW, as instruções trigger-step são executadas para todas as linhas do banco de dados inserido, atualizado ou excluído pela instrução que faz com que o disparador seja acionado, caso a expressão de cláusula WHEN seja avaliada como true.

Caso uma cláusula WHEN seja fornecida, as instruções SQL especificadas como etapas do disparador só são executadas em linhas para as quais a cláusula WHEN é verdadeira. Caso nenhuma cláusula WHEN seja fornecida, as instruções SQL são executadas em todas as linhas.

No corpo de um disparador, (a cláusula trigger-action) os valores anteriores e posteriores à alteração da tabela afetada estão disponíveis usando os nomes de tabela especiais OLD e NEW. A estrutura das tabelas OLD e NEW corresponde à estrutura da tabela na qual o disparador é criado. A tabela OLD contém todas as linhas modificadas ou excluídas pela instrução de acionamento, no estado anterior às operações da instrução de acionamento. A tabela NEW contém todas as linhas modificadas ou criadas pela instrução de acionamento, no estado posterior às operações da instrução de acionamento. A cláusula WHEN e as instruções trigger-step podem acessar valores na linha que está sendo inserida, excluída ou atualizada usando referências da forma NEW.column-name e OLD.column-name, em que column-name é o nome de uma coluna da tabela à qual o disparador está associado. A disponibilidade das referências de tabela OLD e NEW depende do tipo de database-event tratado pelo disparador:

- INSERT – Referências NEW são válidas
- UPDATE – Referências NEW e OLD são válidas
- DELETE – Referências OLD são válidas

O tempo especificado (BEFORE, AFTER ou INSTEAD OF) determina quando as instruções trigger-step são executadas em relação à inserção, modificação ou remoção da linha associada. Uma cláusula ON CONFLICT pode ser especificada como parte de uma instrução UPDATE ou INSERT em uma trigger-step. No entanto, caso uma cláusula ON CONFLICT seja especificada como parte da instrução, o que faz com que o disparador seja acionado, é usada essa política de tratamento de conflitos.

Além dos disparadores de tabela, um disparador INSTEAD OF pode ser criado em uma exibição. Caso um ou mais disparadores INSTEAD OF INSERT, INSTEAD OF DELETE ou INSTEAD OF UPDATE sejam definidos em uma exibição, isso não é considerado erro para executar o tipo associado de instrução (INSERT, DELETE ou UPDATE) na exibição. Nesse caso, executar uma INSERT, DELETE ou UPDATE na exibição faz com que os disparadores associados sejam acionados. Como o disparador é INSTEAD OF, as tabelas subjacentes à exibição não são modificadas pela instrução que faz com que o disparador seja acionado. No entanto, os disparadores podem ser usados para realizar operações de modificação nas tabelas subjacentes.

Há algo importante para se ter em mente durante a criação de um disparador em uma tabela com uma coluna INTEGER PRIMARY KEY. Caso um disparador BEFORE modifique a coluna INTEGER PRIMARY KEY de uma linha a ser atualizada pela instrução que faz com que o disparador seja acionado, a atualização não ocorre. Uma solução é criar a tabela com uma coluna PRIMARY KEY e não uma coluna INTEGER PRIMARY KEY.

Um disparador pode ser removido usando a instrução DROP TRIGGER. Quando uma tabela ou exibição é ignorada, todos os disparadores associados também são ignorados automaticamente.

Função RAISE()

Uma função SQL RAISE() especial pode ser usada em uma instrução trigger-step de um disparador. Essa função tem a seguinte sintaxe:

```
raise-function  ::=  RAISE ( ABORT, error-message ) |
                     RAISE ( FAIL, error-message ) |
                     RAISE ( ROLLBACK, error-message ) |
                     RAISE ( IGNORE )
```

Quando uma das três primeiras formas é chamada durante a execução do disparador, a ação de processamento ON CONFLICT especificada (ABORT, FAIL ou ROLLBACK) é realizada, e a execução da instrução atual é concluída. Como ROLLBACK é considerada uma falha na execução da instrução, a instância de SQLStatement cujo método execute() estava sendo realizado despacha um evento error (SQLErrorEvent.ERROR). O objeto SQLError na propriedade error do objeto do evento despachado tem o conjunto de propriedades details definido como a error-message especificada na função RAISE().

Quando RAISE(IGNORE) é chamada, o restante do disparador atual, a instrução que fez com que o disparador fosse executado e todos os disparadores subsequentes que seriam executados são abandonados. Nenhuma alteração feita no banco de dados é revertida. Caso a instrução que fez com que o disparador fosse executado seja ela própria parte de um disparador, esse programa disparador reinicia a execução no início da próxima etapa. Para obter mais informações sobre os algoritmos para resolução de conflitos, consulte a seção ON CONFLICT (algoritmos em conflito).

DROP TRIGGER

A instrução DROP TRIGGER remove um disparador criado pela instrução CREATE TRIGGER.

```
sql-statement  ::=  DROP TRIGGER [IF EXISTS] [database-name.] trigger-name
```

O disparador é excluído do banco de dados. Observe que os disparadores são ignorados automaticamente quando a tabela associada é ignorada.

Instruções especiais e cláusulas

Esta seção descreve várias cláusulas que são extensões do SQL fornecidas pelo tempo de execução, bem como dois elementos de linguagem que podem ser usados em muitas instruções, comentários e expressões.

COLLATE

A cláusula COLLATE é usada em instruções SELECT, CREATE TABLE e CREATE INDEX para especificar o algoritmo de comparação usado durante a comparação ou a classificação dos valores.

```
sql-statement   ::=  COLLATE collation-name
collation-name  ::=  BINARY | NOCASE
```

O tipo de intercalação padrão de colunas é BINARY. Quando a intercalação BINARY é usada com valores da classe de armazenamento TEXT, a intercalação binária é realizada comparando-se os bytes na memória que representam o valor, independentemente da codificação do texto.

A sequência de intercalação NOCASE só é aplicada a valores da classe de armazenamento TEXT. Quando usada, a intercalação NOCASE realiza uma comparação sem diferenciação entre maiúsculas e minúsculas.

Nenhuma sequência de intercalação é usada em classes de armazenamento do tipo NULL, BLOB, INTEGER ou REAL.

Para usar um tipo de intercalação que não seja BINARY com uma coluna, uma cláusula COLLATE deve ser especificada como parte da definição da coluna na instrução CREATE TABLE. Sempre que dois valores TEXT são comparados, uma sequência de intercalação é usada para determinar os resultados da comparação de acordo com as seguintes regras:

- Em operadores de comparação binária, caso um dos operandos seja uma coluna, o tipo de intercalação padrão da coluna determina a sequência de intercalação usada na comparação. Caso ambos os operandos sejam colunas, o tipo de intercalação do operando à esquerda determina a sequência de intercalação usada. Caso nenhum dos operandos seja uma coluna, é usada a sequência de intercalação BINARY.
- O operador BETWEEN...AND equivale ao uso de duas expressões com os operadores >= e <=. Por exemplo, a expressão x BETWEEN y AND z é equivalente a x >= y AND x <= z. Consequentemente, o operador BETWEEN...AND segue a regra anterior para determinar a sequência de intercalação.
- O operador IN se comporta como o operador = com o objetivo de determinar a sequência de intercalação a ser usada. Por exemplo, a sequência de intercalação usada na expressão x IN (y, z) é o tipo de intercalação padrão x caso x seja uma coluna. Do contrário, é usada a intercalação BINARY.
- Uma cláusula ORDER BY que faz parte de uma instrução SELECT pode ser atribuída explicitamente a uma sequência de intercalação a ser usada na operação de classificação. Nesse caso, a sequência de intercalação explícita é usada sempre. Do contrário, caso a expressão classificada por uma cláusula ORDER BY seja uma coluna, o tipo de intercalação padrão é usado para determinar a ordem de classificação. Caso a expressão não seja uma coluna, é usada a sequência de intercalação BINARY.

EXPLAIN

O modificador de comando EXPLAIN é uma extensão não padrão do SQL.

```
sql-statement  ::=  EXPLAIN sql-statement
```

Caso a palavra-chave EXPLAIN seja exibida antes de qualquer outra instrução SQL, em vez de efetivamente executar o comando, o resultado informa a sequência das instruções da máquina virtual que seriam usadas para executar o comando, se a palavra-chave EXPLAIN não estivesse presente. O recurso EXPLAIN é um recurso avançado e permite aos desenvolvedores alterar o texto da instrução SQL em uma tentativa de otimizar o desempenho ou depurar uma instrução que não parece estar funcionando corretamente.

ON CONFLICT (algoritmos em conflito)

A cláusula ON CONFLICT não é um comando SQL separado. Trata-se de uma cláusula não padrão que pode ser exibida em muitos outros comandos SQL.

```
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-clause     ::=  OR conflict-algorithm
conflict-algorithm  ::=  ROLLBACK |
                         ABORT |
                         FAIL |
                         IGNORE |
                         REPLACE
```

A primeira forma da cláusula ON CONFLICT, usando as palavras-chave ON CONFLICT, é usada em uma instrução CREATE TABLE. Em uma instrução INSERT ou UPDATE, é usada a segunda forma, com ON CONFLICT substituída por OR para fazer a sintaxe parecer mais natural. Por exemplo, em vez de INSERT ON CONFLICT IGNORE, a instrução se torna INSERT OR IGNORE. Embora as palavras-chave sejam diferentes, o significado da cláusula é o mesmo nas duas formas.

A cláusula ON CONFLICT especifica o algoritmo usado para resolver conflitos de restrição. Os cinco algoritmos são ROLLBACK, ABORT, FAIL, IGNORE e REPLACE. O algoritmo padrão é ABORT. A seguir, uma explicação dos cinco algoritmos de conflito:

ROLLBACK Quando ocorre uma violação de restrição, ocorre uma ROLLBACK imediata, encerrando a transação atual. O comando é abortado e a ocorrência de SQLStatement despacha um erro error. Caso não haja nenhuma transação ativa (outra que não seja a implícita criada em todos os comandos), esse algoritmo funciona da mesma forma que ABORT.

ABORT Quando ocorre uma violação de restrição, o comando restaura todas as alterações anteriores que possa ter feito e a ocorrência de SQLStatement gera um evento error. Como nenhuma ROLLBACK é executada, as alterações feitas em comandos anteriores em uma transação são preservadas. ABORT é o comportamento padrão.

FAIL Quando ocorre uma violação de restrição, o comando é abortado e a SQLStatement gera um evento error. No entanto, todas as alterações feitas no banco de dados pela instrução antes de encontrar a violação de restrição são preservadas, e não restauradas. Por exemplo, caso uma instrução UPDATE encontre uma violação de restrição na 100ª linha que ela tenta atualizar, as alterações iniciais feitas nas 99 linhas são preservadas, mas as feitas nas linhas 100 e posteriores não ocorrem.

IGNORE Quando ocorre uma violação de restrição, uma linha que contém a violação de restrição não é inserida ou alterada. Além de essa linha ser ignorada, o comando continua sendo executado normalmente. Outras linhas anteriores e posteriores à linha que apresentava a violação de restrição continuavam sendo inseridas ou atualizadas normalmente. Nenhum erro é retornado.

REPLACE Quando ocorre uma violação de restrição UNIQUE, as linhas preexistentes que estão causando a violação de restrição são removidas antes de inserir ou atualizar a linha atual. Consequentemente, a inserção ou a atualização sempre ocorre e o comando continua sendo executado normalmente. Nenhum erro é retornado. Caso ocorra uma violação de restrição NOT NULL, o valor NULL é substituído pelo valor padrão da coluna. Caso a coluna não tenha nenhum valor padrão, é usado o algoritmo ABORT. Caso ocorra uma violação de restrição CHECK, é usado o algoritmo IGNORE. Quando essa estratégia de resolução de conflitos exclui linhas para atender a uma restrição, ela não invoca disparadores de exclusão nessas linhas.

O algoritmo especificado na cláusula OR de uma instrução INSERT ou UPDATE substitui todos os algoritmos especificados em uma instrução CREATE TABLE. Caso nenhum algoritmo seja especificado na instrução CREATE TABLE ou na instrução INSERT ou UPDATE em execução, é usado o algoritmo ABORT.

REINDEX

O comando REINDEX é usado para excluir e recriar um ou mais índices. Esse comando é útil quando a definição de uma sequência de intercalação é alterada.

```
sql-statement  ::=  REINDEX collation-name
sql-statement  ::=  REINDEX [database-name .] ( table-name | index-name )
```

Na primeira forma, todos os índices em todos os bancos de dados anexados que usam a sequência de intercalação nomeada são recriados. Na segunda forma, quando é especificada table-name, todos os índices associados à tabela são recriados. Caso uma index-name seja fornecida, apenas o índice especificado é excluído e recriado.

COMMENTS

Comentários não são comandos SQL, mas podem ocorrer em consultas SQL. Eles são tratados como espaço em brando pelo tempo de execução. Eles podem começar em qualquer lugar onde haja um espaço em branco, inclusive dentro de expressões que se estendam por várias linhas.

```
comment             ::=  single-line-comment |
                         block-comment
single-line-comment ::=  -- single-line
block-comment       ::=  /* multiple-lines or block [*/]
```

Um comentário com uma única linha é indicado por dois traços. Ele só se estende até o final da linha atual.

Comentários em bloco podem se estender por um número variado de linhas ou ser incorporados a uma única linha. Caso não haja nenhum delimitador de encerramento, um comentário em bloco se estende até o final da entrada. Essa situação não é tratada como um erro. Uma nova instrução SQL pode começar em uma linha após o término do comentário em bloco. Os comentários em bloco podem ser incorporados em qualquer lugar onde haja um espaço em branco, inclusive expressões internas e no meio de outras instruções SQL. Os comentários em bloco não são aninhados. Comentários em linha única dentro de um comentário em bloco são ignorados.

EXPRESSIONS

As expressões são subcomandos dentro de outros blocos SQL. A seguir, a descrição da sintaxe válida de uma expressão dentro de uma instrução SQL:

```
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
```

Uma expressão é uma combinação de valores e operadores que podem ser resolvidos como um valor único. As expressões podem ser divididas em dois tipos gerais, dependendo de sua resolução como um valor booliano (true ou false) ou como um valor não booliano.

Em várias situações comuns, inclusive em uma cláusula WHERE, HAVING, a expressão ON em uma cláusula JOIN e uma expressão CHECK, a expressão deve ser resolvida como um valor booliano. Os tipos de expressões a seguir atendem a essa condição:

- ISNULL
- NOTNULL
- IN ()
- EXISTS ()
- LIKE
- GLOB
- Determinadas funções
- Determinados operadores (mais especificamente operadores de comparação)

Valores literais

Um valor numérico literal é escrito como um número inteiro ou um número de ponto flutuante. A notação científica é suportada. O quantificador . (ponto final) é sempre usado como a casa decimal.

Uma string literal é indicada colocando a string entre aspas simples '. Para incluir uma aspa simples dentro de uma string, coloque duas aspas simples em sequência como este exemplo:

Um literal booliano é indicado pelo valor true ou false. Os valores boolianos literais são usados com o tipo de dados da coluna Booliana.

Um literal de BLOB é uma string literal contendo dados hexadecimais e após um único caractere x ou X como, por exemplo, X'53514697465'.

Um valor literal também pode ser o token NULL.

Nome da coluna

Um nome de coluna pode ser qualquer um dos nomes definidos na instrução CREATE TABLE ou em um dos seguintes identificadores especiais: Esses identificadores especiais descrevem a chave de inteiro aleatória exclusiva (a chave de linha) associada a todas as linhas de cada tabela. Os identificadores especiais só se referem à chave de linha caso a instrução CREATE TABLE não defina uma coluna real com o mesmo nome. As chaves de linha se comportam como colunas somente leitura. Uma chave de linha pode ser usada em qualquer lugar onde uma coluna normal possa ser usada, exceto pelo fato de não ser possível alterar o valor de uma chave de linha em uma instrução UPDATE ou INSERT. A instrução SELECT * FROM table não inclui a chave de linha no conjunto de resultados.

Instrução SELECT

Uma instrução SELECT pode ser exibida em uma expressão como o operando à direita do operador IN, como quantidade escalar (um único valor de resultado), ou como o operando de um operador EXISTS. Quando usada como quantidade escalar ou o operando de um operador IN, a SELECT só pode ter uma única coluna no resultado. Uma instrução SELECT composta (conectada com palavras-chave como UNION ou EXCEPT) é permitida. Com o operador EXISTS, as colunas do conjunto de resultados de SELECT são ignoradas e a expressão retorna TRUE caso haja uma ou mais linhas e FALSE caso o conjunto de resultados seja vazio. Caso nenhum termo da expressão SELECT se refira ao valor na consulta, a expressão é avaliada uma vez antes de qualquer outro processamento e o resultado é reutilizado conforme necessário. Caso a expressão SELECT não contenha variáveis da consulta externa, conhecida como subconsulta correlacionada, a SELECT é reavaliada sempre que necessário.

Quando uma SELECT for o operando à direita do operador IN, o operador IN retorna TRUE caso o resultado do operando à esquerda seja igual a qualquer um dos valores no conjunto de resultados da instrução SELECT. O operador IN pode ser precedido pela palavra-chave NOT para inverter o sentido do teste.

Quando uma SELECT é exibida dentro de uma expressão, mas não é operando à direita de um operador IN, a primeira linha do resultado da SELECT se torna o valor usado na expressão. Caso a SELECT produza mais de uma linha de resultado, todas as linhas depois dela são ignoradas. Caso a SELECT não produza nenhuma linha, o valor da SELECT é NULL.

expressão CAST

Uma expressão CAST altera o tipo de dados do valor especificado para o especificado. O tipo especificado pode ser um nome de tipo não vazio válido para o tipo em uma definição de coluna de uma instrução CREATE TABLE. Suporte ao tipo de dados

Elementos de expressão adicionais

Os elementos SQL a seguir também podem ser usados em expressões:

- Funções internas: funções agregadas, funções escalares e funções de formatação de data/hora
- Operadores
- Parâmetros

funções internas

As funções internas se encaixam em três categorias principais:

- funções agregadas
- funções escalares
- Funções de data e hora

Além dessas funções, existe uma função especial RAISE() usada para fornecer a notificação de um erro na execução de um disparador. Essa função só pode ser usada no corpo de uma instrução CREATE TRIGGER. Para obter informações sobre a função RAISE(), consulte CREATE TRIGGER > RAISE().

Assim como todas as palavras-chave em SQL, os nomes da função não diferenciam maiúsculas de minúsculas.

funções agregadas

As funções agregadas realizam operações em valores de várias linhas. Essas funções são especialmente usadas em instruções SELECT com a cláusula GROUP BY.

AVG(X)	Retorna o valor médio de todo X que não seja NULL em um grupo. Os valores de string e BLOB que não se parecem com números são interpretados como 0. O resultado de AVG() é sempre um valor de ponto flutuante, mesmo que todas as entradas sejam inteiros.
COUNT(X) COUNT(*)	A primeira forma retorna uma contagem do número de vezes em que X não é NULL em um grupo. A segunda forma (com o argumento *) retorna o número total de linhas no grupo.
MAX(X)	Retorna o valor máximo de todos os valores no grupo. A ordem de classificação comum é usada para determinar o máximo.
MIN(X)	Retorna o valor mínimo que não seja NULL de todos os valores no grupo. A ordem de classificação comum é usada para determinar o mínimo. Caso todos os valores no grupo sejam NULL, NULL é retornado.
SUM(X) TOTAL(X)	Retorna a soma numérica de todos os valores que não sejam NULL no grupo. Caso todos os valores sejam NULL, SUM() retorna NULL e TOTAL() retorna 0.0. O resultado de SUM() é um valor inteiro caso todas as entradas que não sejam NULL sejam inteiros. Caso alguma entrada SUM() não seja um inteiro e não NULL, SUM() retorna um valor de ponto flutuante. Esse valor pode ser uma aproximação da soma real.

Em qualquer uma das funções agregadas anteriores que usam um único argumento, esse argumento pode ser precedido pela palavra-chave DISTINCT. Nesse caso, os elementos duplicados são filtrados antes de serem passados para a função agregada. Por exemplo, a chamada de função COUNT(DISTINCT x) retorna o número de valores distintos da coluna X e não o número total de valores que não sejam NULL na coluna x.

funções escalares

As funções escalares funcionam em valores uma linha por vez.

ABS(X)	Retorna o valor absoluto do argumento X.
COALESCE(X, Y, ...)	Retorna uma cópia do primeiro argumento que não seja NULL. Caso todos os argumentos sejam NULL, NULL é retornado. Deve haver pelo menos dois argumentos.
GLOB(X, Y)	Essa função é usada para implementar a sintaxe X GLOB Y.
IFNULL(X, Y)	Retorna uma cópia do primeiro argumento que não seja NULL. Caso ambos os argumentos sejam NULL, NULL é retornado. Essa função se comporta da mesma forma que COALESCE().
HEX(X)	O argumento é interpretado como um valor do tipo de armazenamento BLOB. O resultado é um processamento hexadecimal do conteúdo desse valor.
LAST_INSERT_ROWID( )	Retorna o identificador de linha (chave primária gerada) da última linha inserida no banco de dados pela SQLConnection atual. Este valor é o mesmo valor retornado pela propriedade SQLConnection.lastInsertRowID.
LENGTH(X)	Retorna o comprimento da string de X em caracteres.
LIKE(X, Y [, Z])	Essa função é usada para implementar a sintaxe X LIKE Y [ESCAPE Z] de SQL. Caso a cláusula ESCAPE opcional esteja presente, a função é invocada com três argumentos. Do contrário, ela é invocada com apenas dois argumentos.
LOWER(X)	Retorna uma cópia da string X com todos os caracteres convertidos em minúsculas.
LTRIM(X) LTRIM(X, Y)	Retorna uma string formada pela remoção dos espaços à esquerda de X. Caso um argumento Y seja especificado, a função remove todos os caracteres em Y à esquerda de X.
MAX(X, Y, ...)	Retorna o argumento com o valor máximo. Além de números, os argumentos podem ser strings. O valor máximo é determinado pela ordem de classificação definida. Observe que MAX() é uma função simples quando tem 2 ou mais argumentos, mas uma função agregada quando tem um único argumento.
MIN(X, Y, ...)	Retorna o argumento com o valor mínimo. Além de números, os argumentos podem ser strings. O valor mínimo é determinado pela ordem de classificação definida. Observe que MIN() é uma função simples quando tem 2 ou mais argumentos, mas uma função agregada quando tem um único argumento.
NULLIF(X, Y)	Retorna o primeiro argumento caso os argumentos sejam diferentes; do contrário, ele retorna NULL.
QUOTE(X)	Essa rotina retorna uma string que é o valor do argumento apropriado à inclusão em outra instrução SQL. As strings são colocadas entre aspas simples com escapes em aspas internas conforme necessário. As classes de armazenamento BLOB são codificadas como literais hexadecimais. A função é útil durante a criação de disparadores para implementar a funcionalidade desfazer/refazer.
RANDOM(*)	Gera um inteiro pseudoaleatório entre -9223372036854775808 e 9223372036854775807. Esse valor aleatório não é criptográfico.
RANDOMBLOB(N)	Retorna um BLOB de N bytes que contém bytes pseudoaleatórios. N deve ser um inteiro positivo. Esse valor aleatório não é criptográfico. Caso o valor de N seja negativo, é retornado um único byte.
ROUND(X) ROUND(X, Y)	Arredonda o número X para Y dígitos à direita da casa decimal. Caso o argumento Y seja omitido, é usado 0.
RTRIM(X)	Retorna uma string formada pela remoção dos espaços à direita de X. Caso um argumento Y seja especificado, a função remove todos os caracteres em Y à direita de X.
SUBSTR(X, Y, Z)	Gera uma subsequência de caracteres da sequência de entrada X que começa com o Yş caractere e que tem Z caracteres de comprimento. O caractere mais à esquerda de X é a posição de índice 1. Se Y for negativo, o primeiro caractere da subsequência de caracteres é encontrado contando-se a partir da direita e não da esquerda.
TRIM(X) TRIM(X, Y)	Retorna uma string formada pela remoção dos espaços à direita de X. Caso um argumento Y seja especificado, a função remove todos os caracteres em Y à direita de X.
TYPEOF(X)	Retorna o tipo da expressão X. Os valores de retorno possíveis são 'null', 'integer', 'real', 'text' e 'blob'. Suporte ao tipo de dados
UPPER(X)	Retorna uma cópia da string de entrada X convertida em letras todas em maiúsculas.
ZEROBLOB(N)	Retorna um BLOB que contém N bytes de 0x00.

funções de formatação de data e hora

As funções de formatação de data e hora são um grupo de funções escalares usadas para criar dados de data e hora formatados. Observe que essas funções operam e retornam valores de string e número. Essas funções não devem ser usadas com o tipo de dados DATE. Caso você use essas funções nos dados de uma coluna cujo tipo de dados declarado é DATE, elas não se comportam como esperado.

DATE(T, ...)	A função DATE() retorna uma string contendo a data neste formato: YYYY-MM-DD. O primeiro parâmetro (T) especifica uma string de hora do formato encontrado em Formatos de hora. É possível especificar um número qualquer de modificadores após a string de hora. Esses modificadores podem ser encontrados em Modificadores.
TIME(T, ...)	A função TIME() retorna uma string que contém a hora como HH:MM:SS. O primeiro parâmetro (T) especifica uma string de hora do formato encontrado em Formatos de hora. É possível especificar um número qualquer de modificadores após a string de hora. Esses modificadores podem ser encontrados em Modificadores.
DATETIME(T, ...)	A função DATETIME() retorna uma string que contém a data e a hora no formato YYYY-MM-DD HH:MM:SS. O primeiro parâmetro (T) especifica uma string de hora do formato encontrado em Formatos de hora. É possível especificar um número qualquer de modificadores após a string de hora. Esses modificadores podem ser encontrados em Modificadores.
JULIANDAY(T, ...)	A função JULIANDAY()retorna um número que indica quantos dias em relação a Greenwich em 24 de novembro de 4714 A.C. e a data fornecida. O primeiro parâmetro (T) especifica uma string de hora do formato encontrado em Formatos de hora. É possível especificar um número qualquer de modificadores após a string de hora. Esses modificadores podem ser encontrados em Modificadores.
STRFTIME(F, T, ...)	A rotina STRFTIME() retorna a data formatada de acordo com a string do formato especificado como o primeiro argumento F. A string do formato oferece suporte às seguintes substituições: %d - dia do mês %f - segundos fracionários SS.SSS %H - hora 00-24 %j - dia do ano 001-366 %J - número do dia Juliano %m - mês 01-12 %M - minuto 00-59 %s - segundos desde 1970-01-01 %S - segundo 00-59 %w - dias da semana 0-6 (domingo = 0) %W - semana do ano 00-53 %Y - ano 0000-9999 %% - % O segundo parâmetro (T) especifica uma string de hora do formato encontrado em Formatos de hora. É possível especificar um número qualquer de modificadores após a string de hora. Esses modificadores podem ser encontrados em Modificadores.

Formatos de hora

Uma string de hora pode estar em qualquer um dos seguintes formatos:

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 (a data é 2000-01-01)
HH:MM:SS	07:30:59 (a data é 2000-01-01)
HH:MM:SS.SSS	07:30:59:152 (a data é 2000-01-01)
Fazer agora	Data e hora atuais em Universal Coordinated Time (UTC).
DDDD.DDDD	Número do dia Juliano como número de ponto flutuante.

O caractere V nesses formatos é um caractere literal "T" que separa a data e a hora. Os formatos que só incluem uma hora pressupõem a data 2001-01-01.

Modificadores

A string de hora pode ser seguida de zero ou mais modificadores que alteram a data ou a interpretação da data. Os modificadores disponíveis são os seguintes:

NNN days	Número de dias a serem adicionados à hora.
NNN hours	Número de horas a serem adicionadas à hora.
NNN minutes	Número de minutos a serem adicionados à hora.
NNN.NNNN seconds	Número de segundos e milésimos de segundo a serem adicionados à hora.
NNN months	Número de meses a serem adicionados à hora.
NNN years	Número de anos a serem adicionados à hora.
start of month	Alterne a hora para o começo do mês.
start of year	Alterne a hora para o começo do ano.
start of day	Alterne a hora para o começo do dia.
weekday N	Avança a hora para o dia da semana especificado. (0 = domingo, 1 = segunda-feira e assim por diante)
localtime	Converte a data em hora local.
utc	Converte a data em UTC

Operadores

SQL oferece suporte a uma ampla seleção de operadores, inclusive operadores comuns existentes na maior parte das linguagens de programação, bem como vários operadores exclusivos do SQL.

Operadores comuns

Os seguintes operadores binários são permitidos em um bloco SQL, estando listados da maior para a menor precedência:

```
*     /     %
+     -
<<  >>  &  |
<  >=    >  >=
=     ==    !=    <>  IN
AND
OR
```

Os operadores de prefixo unários para os quais há suporte são:

```
 !     ~     NOT
```

O operador COLLATE pode ser considerado um operador de sufixo unário. O operador COLLATE apresenta a maior precedência. Ele sempre é mais vinculado do que qualquer operador unário de prefixo ou operador binário.

Observe que existem duas variações dos operadores de igualdade e de não-igualdade. A igualdade pode ser = ou ==. O operador de não igualdade pode ser != ou <>.= ou <>.

O operador || é o operador de concatenação da string—ele une as duas strings dos operandos.

O operador % fornece o restante do módulo do operando à esquerda no operando à direita.

O resultado de qualquer operador binário é um valor numérico, exceto pelo operador de concatenação || que fornece um resultado de string.

Operadores SQL

LIKE

O operador LIKE realiza uma comparação de padrões.

```
expr     ::=  (column-name | expr) LIKE pattern
pattern  ::=  '[ string | % | _ ]'
```

O operando à direita do operador LIKE contém o padrão e o operando à esquerda contém a string de acordo com o padrão. Um símbolo de porcentagem (%) no padrão é um caractere curinga—ele corresponde a qualquer sequência de zero ou mais caracteres na string. Um sublinhado (_) no padrão corresponde a um único caractere da string. Qualquer outro caractere se compara ou o equivalente em maiúsculas/minúsculas, ou seja, são feitas as correspondências de maneira sem diferenciação de maiúsculas e minúsculas. o mecanismo de banco de dados só compreende maiúsculas/minúsculas em caracteres Latinos de 7 bits. Consequentemente, o operador LIKE diferencia maiúsculas e minúsculas para caracteres iso8859 de 8 bits ou caracteres UTF-8. Por exemplo, a expressão 'a' LIKE 'A' é TRUE, mas 'æ' LIKE 'Æ' é FALSE). A diferenciação entre maiúsculas e minúsculas para caracteres Latin pode ser alterada usando a propriedade SQLConnection.caseSensitiveLike.

Caso a cláusula opcional ESCAPE esteja presente, a expressão após a palavra-chave ESCAPE deve ser avaliada como uma string que consiste em um único caractere. Esse caractere pode ser usado no padrão LIKE para corresponder aos caracteres de porcentagem literal ou sublinhado. O caractere de escape seguido de um símbolo de porcentagem, sublinhado ou ele próprio compara um símbolo de porcentagem, sublinhado ou caractere de escapa na string, respectivamente.

GLOB

O operador GLOB é semelhante a LIKE, mas usa a sintaxe globbing do arquivo Unix nos curingas. Diferentemente de LIKE, GLOB diferencia maiúsculas de minúsculas.

IN

O operador IN calcula se o operando à esquerda é igual a um dos valores no operando à direita (um conjunto de valores entre parênteses).

```
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]*
```

O operando à direita pode ser um conjunto de valores literais separados por vírgulas ou o resultado de uma instrução SELECT. Consulte as instruções SELECT em expressões para obter uma explicação e saber as limitações quanto ao uso de uma instrução SELECT como operando à direita do operador IN.

BETWEEN...AND

O operador BETWEEN...AND equivale ao uso de duas expressões com os operadores >= e <=. Por exemplo, a expressão x BETWEEN y AND z é equivalente a x >= y AND x <= z.

NOT

NOT é um operador de negação. Os operadores GLOB, LIKE, e IN podem ser precedidos pela palavra-chave NOT para inverter o sentido do teste (em outras palavras, para verificar se um valor não corresponde ao padrão indicado).

Parâmetros

Um parâmetro especifica um alocador de espaço na expressão para um valor literal preenchido durante o tempo de execução atribuindo um valor à matriz associativa SQLStatement.parameters. Os parâmetros podem ter três formas:

?	Um ponto de interrogação indica um parâmetro indexado. Os parâmetros recebem valores de índice numérico (com base em zero) de acordo com a ordem na instrução.
:AAAA	Uma vírgula seguida de um nome de identificador mantém um ponto no parâmetro nomeado com o nome AAAA. Os parâmetros nomeados também são numerados de acordo com sua ordem na instrução SQL. Para evitar confusão, é melhor evitar a mistura de parâmetros nomeados e numerados.
@AAAA	Um sinal de arroba é equivalente a uma vírgula.

Recursos SQL para os quais não há suporte

Esta é uma lista dos elementos SQL padrão para os quais não há suporte no Adobe AIR:

Restrições de FOREIGN KEY As restrições de FOREIGN KEY são analisadas, mas não são exigidas.

Disparadores Os disparadores FOR EACH STATEMENT não são permitidos (todos os disparadores devem ser FOR EACH ROW). Os disparadores INSTEAD OF não podem ser usados em tabelas (os disparadores INSTEAD OF só podem ser usados em modos de exibição). Disparadores recursivos — disparadores que se acionam automaticamente — não são suportados.

ALTER TABLE Apenas as variantes RENAME TABLE e ADD COLUMN do comando ALTER TABLE podem ser usadas. Outros tipos de operações ALTER TABLE como, por exemplo, DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT etc. são ignorados.

Transações aninhadas Apenas uma única transação ativa é permitida.

RIGHT e FULL OUTER JOIN RIGHT OUTER JOIN ou FULL OUTER JOIN não são permitidos.

VIEW Atualizável Um modo de exibição é exclusivo para leitura. Você talvez não execute uma instrução DELETE, INSERT ou UPDATE em uma exibição. Há suporte para um disparador INSTEAD OF acionado diante de uma tentativa de uso de DELETE, INSERT ou UPDATE em uma exibição, podendo ser usado para atualizar tabelas de suporte no corpo do disparador.

GRANT e REVOKE m banco de dados é um arquivo em disco comum; as únicas permissões de acesso que podem ser aplicadas são as permissões normais de acesso ao arquivo do sistema operacional subjacente. Os comandos GRANT e REVOKE normalmente encontrados em RDBMSs de cliente/servidor não são implementados.

Há suporte aos seguintes elementos SQL e os recursos do SQLite em algumas implementações de SQLite, mas não no Adobe AIR. Grande parte dessa funcionalidade está disponível por meio de métodos da classe SQLConnection:

Elementos SQL relacionados à transação (BEGIN, END, COMMIT, ROLLBACK) Essa função é disponibilizada pelos métodos relacionados à transação da classe SQLConnection: SQLConnection.begin(), SQLConnection.commit() e SQLConnection.rollback().

ANALYZE Essa função é disponibilizada pelo método SQLConnection.analyze().

ATTACH Essa função é disponibilizada pelo método SQLConnection.attach().

COPY Esta instrução não é suportada.

CREATE VIRTUAL TABLE Esta instrução não é suportada.

DETACH Essa função é disponibilizada pelo método SQLConnection.detach().

PRAGMA Esta instrução não é suportada.

VACUUM Essa função é disponibilizada pelo método SQLConnection.compact().

O acesso à tabela do sistema não está disponível As tabelas do sistema, inclusive sqlite_master e outras com o prefixo sqlite_, não estão disponíveis nas instruções SQL. O tempo de execução inclui uma API de esquema que fornece uma forma orientada a objetos para acessar os dados do esquema. Para obter mais informações, consulte o método SQLConnection.loadSchema().

Funções de expressão regular (MATCH() e REGEX()) Essas funções não estão disponíveis em instruções SQL.

A seguinte funcionalidade é diferente em relação às muitas implementações do SQLite e do Adobe AIR:

Parâmetros de instrução indexados Em muitas implementações, os parâmetros de instrução indexados se baseiam em um. No entanto, nos parâmetros de instrução indexados do Adobe AIR se baseiam em zero (ou seja, o primeiro parâmetro recebe o índice 0, o segundo parâmetro recebe o índice 1 e assim por diante.

Definições da coluna INTEGER PRIMARY KEY Em muitas implementações, apenas as colunas definidas exatamente como INTEGER PRIMARY KEY são usadas como a coluna real de chave primária de uma tabela. Nessas implementações, usar outro tipo de dados que normalmente é sinônimo de INTEGER (como int) não faz com que a coluna seja usada como chave primária interna. Entretanto, no Adobe AIR, o tipo de dados int (e outros sinônimos de INTEGER) é considerado exatamente equivalente a INTEGER. Portanto, uma coluna definida como int PRIMARY KEY é usada como chave primária interna de uma tabela. Para obter mais informações, consulte as seções CREATE TABLE e Afinidade de colunas.

Recursos SQL adicionais

Os seguintes tipos de afinidade de colunas não podem ser usados por padrão no SQLite, mas são reconhecidos pelo Adobe AIR (perceba que, como todas as palavras-chaves do SQL, esses nomes de tipos de dados não fazem distinção entre maiúsculas e minúsculas):

Booliano correspondente à classe Boolean.

Date correspondente à classe Date.

int correspondente à classe int (equivalente à afinidade de coluna INTEGER).

Number correspondente à classe Number (equivalente à afinidade de coluna REAL).

Objeto correspondente à classe Object ou a qualquer subclasse que possa ser serializada e desserializada usando AMF3. (Isso inclui a maioria das classes, inclusive classes personalizadas, mas exclui algumas, dentre elas objetos de exibição e objetos que incluem objetos de exibição como propriedades.)

String correspondente à classe String (equivalente à afinidade de coluna TEXT).

XML correspondente à classe XML do ActionScript (E4X).

XMLList correspondente à classe XMLList do ActionScript (E4X).

Por padrão, não há suporte aos valores literais no SQLite, embora haja no Adobe AIR:

true usado para representar o valor booliano literal true, para trabalhar com colunas BOOLEAN.

false usado para representar o valor booliano literal false, para trabalhar com colunas BOOLEAN.

Suporte ao tipo de dados

Ao contrário da maioria dos bancos de dados SQL, o mecanismo SQL do Adobe AIR não exige ou impõe que essas colunas de tabela contenham valores de um determinado tipo. Na verdade, o tempo de execução usa dois conceitos, classes de armazenamento e afinidade de coluna, para controlar tipos de dados. Esta seção descreve as classes de armazenamento e a afinidade de coluna, bem como a forma com que as diferenças no tipo de dados são resolvidas em várias condições:

- "Classes de armazenamento" na página 1171
- "Afinidade de coluna" na página 1172
- "Tipos de dados e operadores de comparação" na página 1174
- "Tipos de dados e operadores matemáticos" na página 1175
- "Tipos de dados e classificação" na página 1175
- "Tipos de dados e agrupamento" na página 1175
- "Tipos de dados e instruções SELECT compostas" na página 1175

Classes de armazenamento

As classes de armazenamento representam os tipos de dados reais usados para armazenar valores em um banco de dados. As seguintes classes de armazenamento são usadas pelo banco de dados:

NULL O valor é um valor NULL.

INTEGER O valor é um inteiro assinado.

REAL O valor é um número de ponto flutuante.

TEXT O valor é uma string de texto (limitada a 256 MB).

BLOB O valor é um BLOB (objeto binário grande); em outras palavras, dados binários brutos (limitados a 256 MB).

Todos os valores fornecidos ao banco de dados como literais incorporados em uma instrução SQL ou valores associados usando parâmetros a uma instrução SQL preparada são atribuídos a uma classe de armazenamento antes da execução da instrução SQL.

Os literais que fazem parte de uma instrução SQL são atribuídos à classe de armazenamento TEXT, caso estejam entre aspas simples ou duplas, INTEGER caso o literal seja especificado como um número sem aspas e casa decimal ou expoente, REAL caso o literal seja um número sem aspas com casa decimal ou expoente e NULL caso o valor seja NULL. Os literais com um BLOB da classe de armazenamento são especificados usando a notação X'ABCD'. Para obter mais informações, consulte Valores literais em expressões.

Valores fornecidos como parâmetros usando a matriz associativa SQLStatement.parameters são atribuídos à classe de armazenamento que mais corresponda ao tipo de dados nativos associado. Por exemplo, os valores int são associados como uma classe de armazenamento INTEGER, os valores Number recebem a classe de armazenamento REAL, os valores String recebem a classe de armazenamento TEXT e os objetos ByteArray recebem a classe de armazenamento BLOB.

Afinidade de coluna

A *afinidade* de uma coluna é o tipo recomendado para dados armazenados nessa coluna. Quando um valor é armazenado em uma coluna (por meio de uma instrução INSERT ou UPDATE), o tempo de execução tenta converter esse valor do tipo de dados na afinidade especificada. Por exemplo, caso um valor Date (uma ocorrência de Date em ActionScript ou JavaScript) seja inserido em uma coluna cuja afinidade é TEXT, o valor Date é convertido na representação String (equivalente à chamada do método toString() do objeto) antes do armazenamento no banco de dados. Caso o valor não possa ser convertido na afinidade especificada, ocorre um erro e a operação não é realizada. Quando um valor é recuperado do banco de dados usando uma instrução SELECT, ele é retornado como uma instância da classe correspondente à afinidade, independentemente de ser convertido de um tipo de dados diferente quando armazenado.

Caso uma coluna aceite valores NULL, o valor ActionScript ou JavaScript null pode ser usado como um valor de parâmetro para armazenar NULL na coluna. Quando um valor da classe de armazenamento NULL é recuperado em uma instrução SELECT, ele sempre é retornado como o valor de ActionScript ou JavaScript null, independentemente da afinidade da coluna. Caso uma coluna aceite valores NULL, sempre verifique valores recuperados da coluna para determinar se eles são null antes de tentar projetar os valores como um tipo que não possa ser nulo (como, por exemplo, Number ou Boolean).

Cada coluna do banco de dados é atribuída a uma das seguintes afinidades de tipo:

- TEXT (ou String)
- NUMERIC
- INTEGER (ou int)
- REAL (ou Number)
- Booliano
- Date
- XML
- XMLLIST
- Objeto
- NONE

TEXT (ou String)

Uma coluna com afinidade TEXT ou String armazena todos os dados que usam classes de armazenamento NULL, TEXT ou BLOB. Caso sejam inseridos dados numéricos em uma coluna com afinidade TEXT, eles são convertidos em forma de texto antes de serem armazenados.

NUMERIC

Uma coluna com afinidade NUMERIC contém valores que usam classes de armazenamento NULL, REAL ou INTEGER. Quando dados de texto são inseridos em uma coluna NUMERIC, é feita uma tentativa de convertê-los em um inteiro ou número real antes de serem armazenados. Caso haja êxito na conversão, o valor é armazenado usando a classe de armazenamento INTEGER ou REAL (por exemplo, um valor igual a '10.05' é convertido na classe de armazenamento REAL antes de ser armazenado). Caso a conversão não possa ser realizada, ocorre um erro. não é feita nenhuma tentativa de converter um valor NULL. Um valor recuperado de uma coluna NUMERIC é retornado como uma ocorrência do tipo numérico mais específico no qual o valor se encaixa. Em outras palavras, caso o valor seja um inteiro positivo ou 0, ele é retornado como uma ocorrência de unit. Caso seja um inteiro negativo, ele é retornado como uma ocorrência de int. Por fim, caso tenha um componente de ponto flutuante (não um inteiro), ele é retornado como uma instância de Number.

INTEGER (ou int)

Uma coluna que usa afinidade INTEGER se comporta da mesma forma que uma coluna com afinidade NUMERIC, exceto por uma coisa. Caso o valor a ser armazenado seja um valor real (como, por exemplo, uma instância de Number) sem nenhum componente de ponto flutuante ou caso o valor seja um valor de texto que possa ser convertido em um valor real sem nenhum componente de ponto flutuante, ele é convertido em um inteiro e armazenado usando a classe de armazenamento INTEGER. Caso seja feita uma tentativa de armazenar um valor real com um componente de ponto flutuante, ocorre um erro.

REAL (ou Number)

Uma coluna com afinidade REAL ou NUMBER se comporta como uma coluna com afinidade NUMERIC, exceto pelo fato de forçar valores inteiros na representação de ponto flutuante. Um valor em uma coluna REAL é sempre retornado do banco de dados como uma instância de Number.

Booliano

Uma coluna com afinidade Boolean armazena valores true ou false. Uma coluna Boolean aceita um valor que é uma ocorrência de Boolean em ActionScript ou JavaScript. Caso o código tente armazenar um valor String, String com comprimento maior que zero é considerada verdadeira e uma String vazia, falsa. Caso o código tente armazenar dados numéricos, qualquer valor que não seja zero é armazenado como true e 0, como false. Quando um valor Boolean é recuperado usando uma instrução SELECT, ele é retornado como uma ocorrência de Boolean. Valores que não sejam NULL são armazenados usando a classe de armazenamento INTEGER (0 para false e 1 para true), sendo convertidos em objetos Boolean quando os dados são recuperados.

Date

Uma coluna com afinidade Date armazena valores de data e hora. Uma coluna Date é projetada para aceitar valores que sejam ocorrências de Date em ActionScript ou JavaScript. Caso seja feita uma tentativa de armazenar um valor String em uma coluna Date, o tempo de execução tenta convertê-lo em uma data do calendário Juliano. Em caso de falha na conversão, ocorre um erro. Caso o código tente armazenar um valor de Number, int ou uint, não é feita nenhuma tentativa de validar os dados, pressupondo que ele seja um valor de data do calendário Juliano válido. Um valor Date recuperado usando uma instrução SELECT é convertido automaticamente em uma ocorrência de Date. Como os valores de Date são armazenados como valores de data do calendário Juliano usando a classe de armazenamento REAL, as operações de classificação e comparação funcionam da forma como você espera.

XML ou XMLList

Uma coluna que usa afinidade XML ou XMLList armazena estruturas XML. Quando o código tenta armazenar dados em uma coluna XML usando um parâmetro SQLStatement, o tempo de execução tenta converter e validar o valor usando a função XML() ou XMLList() em ActionScript. Caso o valor não possa ser convertido em um XML válido, ocorre um erro. Caso a tentativa de armazenar os dados use um valor de texto SQL literal (por exemplo, INSERT INTO (col1) VALUES ('Invalid XML (no closing tag)'), o valor não é analisado ou validado — pressupõe-se que esteja bem-formado. Caso um valor inválido seja armazenado, quando recuperado, ele é retornado como um objeto XML vazio. Dados XML e XMLList são armazenados usando a classe de armazenamento TEXT ou NULL.

Objeto

Uma coluna com afinidade Object armazena objetos complexos em ActionScript ou JavaScript, inclusive ocorrências da classe Object, bem como ocorrências das subclasses de Object como, por exemplo, ocorrências de Array e até mesmo ocorrências de classe personalizada. Os dados da coluna Object são serializados no formato AMF3 e armazenados usando a classe de armazenamento BLOB. Quando recuperado, um valor é desserializado do AMF3 e retornado como uma instância da classe como foi armazenado. Observe que algumas classes ActionScript, mais notadamente objetos de exibição, não podem ser desserializadas como ocorrências do tipo de dados original. Antes de armazenar uma ocorrência de classe personalizada, você deve registrar um alias para a classe usando o método

flash.net.registerClassAlias() (ou em Flex, adicionando metadados [RemoteObject] à declaração da classe). Além disso, antes de recuperar esses dados, você deve registrar o mesmo alias para a classe. Qualquer dado que não possa ser serializado corretamente porque a classe não pode ser desserializada inerentemente ou por conta de um alias de classe ausente ou não correspondente é retornado como um objeto anônimo (uma ocorrência de classe Object) com propriedades e valores correspondentes à ocorrência original de acordo com o armazenamento.

NONE

Uma coluna com afinidade NONE não prefere nenhuma classe de armazenamento em detrimento de outra. Ela não faz nenhuma tentativa de converter dados antes de serem inseridos.

Determinação da afinidade

A afinidade de tipo de uma coluna é determinada pelo tipo declarativo da coluna na instrução CREATE TABLE. Quando o tipo é determinado, as seguintes regras (sem distinção de caixa) se aplicam:

- Caso o tipo de dados da coluna contenha uma das strings CHAR, CLOB, STRI, ou TEXT essa coluna terá afinidade com TEXT/String. Observe que o tipo VARCHAR contém a string CHAR e, por isso, é atribuído à afinidade TEXT.
- Caso o tipo de dados da coluna contenha a string BLOB ou caso nenhum tipo de dados seja especificado, a afinidade da coluna é NONE.
- Caso o tipo de dados da coluna contenha a string XMLL, a coluna tem afinidade XMLList.
- Caso o tipo de dados seja a string XML, a coluna tem afinidade XML.
- Caso o tipo de dados contenha a string OBJE, a coluna tem afinidade Object.
- Caso o tipo de dados contenha a string BOOL, a coluna tem afinidade Boolean.
- Caso o tipo de dados contenha a string DATE, a coluna tem afinidade Date.
- Caso o tipo de dados contenha a string INT (inclusive UINT), ele é atribuído à afinidade INTEGER/int.
- Caso o tipo de dados da coluna contenha uma das strings REAL, NUMB, FLOA ou DOUBa coluna apresenta afinidade REAL/Number.
- Do contrário, a afinidade é NUMERIC.
- Caso uma tabela seja criada usando uma instrução CREATE TABLE t AS SELECT..., todas as colunas não têm nenhum tipo de dados especificado, sendo todas atribuídas à afinidade NONE.

Tipos de dados e operadores de comparação

Há suporte para estes operadores de comparação binária =, <, <=, >= e !=, além de uma operação para testar a associação do conjunto, IN e operador de comparação ternário BETWEEN. Para obter detalhes sobre esses operadores, consulte Operadores.

Os resultados de uma comparação dependem das classes de armazenamento dos dois valores que estão sendo comparados. Durante a comparação entre duas regras, se aplicam as seguintes regras:

- Um valor com classe de armazenamento NULL é considerado inferior a qualquer outro valor (inclusive outro valor com classe de armazenamento NULL).
- Um valor INTEGER ou REAL é inferior a qualquer valor TEXT ou BLOB. Quando INTEGER ou REAL é comparado com outro INTEGER ou REAL, é realizada uma comparação numérica.
- Um valor TEXT é inferior a um valor BLOB. Quando dois valores TEXT são comparados, é realizada uma comparação binária.
- Quando dois valores BLOB são comparados, o resultado é sempre determinado usando uma comparação binária.

O operador ternário BETWEEN é sempre reprojetado como a expressão binária equivalente. Por exemplo, a BETWEEN b AND c é reprojetado como a >= b AND a <= c, mesmo que isso signifique a aplicação de afinidades diferentes a a em cada uma das comparações necessárias à avaliação da expressão.

Expressões do tipo a IN (SELECT b ....) são tratadas pelas três regras enumeradas anteriormente para as comparações binárias, ou seja, de maneira semelhante a a = b. Por exemplo, caso b seja um valor de coluna e a seja uma expressão, a afinidade b é aplicada a a antes da realização de qualquer outra comparação. A expressão a IN (x, y, z) é reprojetada como a = +x OR a = +y OR a = +z. Os valores à direita do operador IN (os valores x, y e z nesse exemplo) são considerados expressões, mesmo que sejam valores de coluna. Caso o valor à esquerda do operador IN seja uma coluna, é usada a afinidade dessa coluna. Caso o valor seja uma expressão, não ocorre nenhuma conversão.

A forma como as comparações são realizadas também pode ser afetada pelo uso de uma cláusula COLLATE. Para obter mais informações, consulte COLLATE.

Tipos de dados e operadores matemáticos

Para cada um dos operadores matemáticos para os quais há suporte, *, /, %, + e -, a afinidade numérica é aplicada a cada operando antes da avaliação da expressão. Caso nenhum operando possa ser convertido na classe de armazenamento NUMERIC com êxito, a expressão é avaliada como NULL.

Quando o operador de concatenação || é usado, cada operando é convertido na classe de armazenamento TEXT antes da avaliação da expressão. Caso nenhum operando possa ser convertido na classe de armazenamento TEXT, o resultado da expressão é NULL. A impossibilidade de conversão do valor pode acontecer em duas situações: caso o valor do operando seja NULL ou caso ele seja um BLOB que contenha uma classe de armazenamento que não seja TEXT.

Tipos de dados e classificação

Quando os valores são classificados por uma cláusula ORDER BY, os valores com uma classe de armazenamento NULL vêm primeiro. Eles são seguidos dos valores INTEGER e REAL dispostos em ordem numérica, seguidos dos valores TEXT em ordem binária ou com base na intercalação especificada (BINARY ou NOCASE). Por fim, vêm os valores BLOB na ordem binária. não ocorre nenhuma conversão de classe antes da classificação.

Tipos de dados e agrupamento

Durante o agrupamento de valores com a cláusula GROUP BY, valores com classes de armazenamento diferentes são consideradas distintas. Uma exceção a isso são os valores INTEGER e REAL, que são considerados iguais caso sejam numericamente equivalentes. Nenhuma afinidade é aplicada a nenhum valor por conta de uma cláusula GROUP BY.

Tipos de dados e instruções SELECT compostas

Os operadores SELECT compostos UNION, INTERSECT e EXCEPT realizam comparações implícitas entre valores. Para que essas comparações sejam realizadas, uma afinidade pode ser aplicada a cada valor. A mesma afinidade, caso haja alguma, é aplicada a todos os valores que podem ser retornados em uma única coluna do conjunto de resultados SELECT. A afinidade aplicada é a afinidade da coluna retornada pela instrução SELECT do primeiro componente que apresenta um valor de coluna (e não outro tipo de expressão) nessa posição. Se, para uma determinada coluna SELECT composta, nenhuma das instruções SELECT do componente retornem um valor de coluna, nenhuma afinidade será aplicada aos valores dessa coluna antes de serem comparados.

Capítulo 67: ID, argumentos e mensagens detalhadas de erro SQL

A classe SQLError representa vários erros que podem ocorrer em meio ao trabalho com um banco de dados SQL local Adobe AIR. Em qualquer exceção, a ocorrência SQLError tem uma propriedade `details`que contém uma mensagem de erro em inglês. Além disso, cada mensagem de erro tem um identificador exclusivo associado disponível na propriedade `detailID` do objeto SQLError. Com a propriedade `detailID` um aplicativo pode identificar a mensagem de erro details específica. O aplicativo pode fornecer texto alternativo para o usuário final no idioma do local. Os valores de argumento da matriz `detailArguments` podem ser substituídos na posição apropriada da string da mensagem de erro. Isso é útil para aplicativos que precisam exibir a mensagem de erro de propriedade `details` relativa a um erro diretamente para os usuários finais em um local específico.

A tabela a seguir contém uma lista dos valores `detailID` e o texto da mensagem de erro em inglês associado. O texto do alocador de espaço nas mensagens indica onde os valores `detailArguments` são substituídos pelo tempo de execução. Essa lista pode ser usada como uma origem para localizar as mensagens de erro que podem ocorrer em operações do banco de dados SQL.

SQLError detailID	Mensagem de erro detalhada em inglês e parâmetros
1001	Conexão fechada.
1102	O banco de dados deve estar aberto para realizar a operação.
1003	%s [,|and %s] nome(s) de parâmetro encontrado(s) na propriedade parameters, mas não no SQL especificado.
1004	Incompatibilidade na contagem de parâmetros. Encontrados valor(es) %d no SQL especificado e %d definido(s) na propriedade parameters. Esperavam-se valores para %s [,|and %s].
1005	Não foi possível ativar a compactação automática.
1006	não foi possível definir o valor pageSize.
1007	O objeto de esquema de nome '%s' e tipo '%s' no banco de dados '%s' não foi encontrado.
1008	O objeto do esquema de nome '%s' no banco de dados '%s' não foi encontrado.
1009	Não foi encontrado nenhum objeto do esquema de tipo '%s' no banco de dados '%s'.
1010	Não foi encontrado nenhum objeto do esquema no banco de dados '%s'.
2001	Estouro de pilha do analisador.
2002	Muitos argumentos na função '%s'
2003	próximo a '%s': erro de sintaxe
2004	já existe outra tabela ou índice com este nome: '%s'
2005	Não há permissão para PRAGMA no SQL.
2006	não é um diretório gravável
2007	Tipo de junção desconhecido ou incompatível:
2008	No momento não há suporte a RIGHT e FULL OUTER JOINs.
2009	Uma junção NATURAL talvez não tenha uma cláusula ON ou USING
2010	Não é possível ter cláusulas ON e USING na mesma junção
2011	não é possível unir usando a coluna '%s' - a coluna não está presente em ambas as tabelas.
2012	Apenas um único resultado permitido para SELECT que é parte de uma expressão
2013	Nenhuma tabela deste tipo: '[%s.]%s'
2014	Nenhuma tabela especificada.
2015	Muitas colunas no conjunto de resultados|muitas colunas em '%s'.
2016	%s Termo ORDER|GROUP BY fora do intervalo - deve estar entre 1 e %d
2017	Muitos termos na cláusula ORDER BY.
2018	%s Termo ORDER BY fora do intervalo - deve estar entre 1 e %d.
2019	O termo ORDER BY %r não corresponde a nenhuma coluna do conjunto de resultados.

2020	A cláusula ORDER BY deve vir depois de ‘%s’ e não antes.
2021	A cláusula LIMIT deve vir depois de ‘%s’ e não antes.
2022	SELECTs à esquerda e à direita de ‘%s’ não têm o mesmo número de colunas de resultado.
2023	Uma cláusula GROUP BY é obrigatória antes de HAVING.
2024	não são permitidas funções agregadas na cláusula GROUP BY.
2025	DISTINCT em agregação deve ser seguida de uma expressão.
2026	Muitos termos em SELECT composta.
2027	muitos termos na cláusula ORDER|GROUP BY.
2028	O disparador temporário talvez não tenha nome qualificado
2030	Disparador ‘%s’ já existente.
2032	não é possível criar o disparador BEFORE|AFTER na exibição: '%s'.
2033	não é possível criar o disparador INSTEAD OF na tabela: '%s'.
2034	Nenhum disparador deste tipo: '%s'.
2035	não há suporte para disparadores recursivos (‘%s’).
2036	Nenhuma coluna deste tipo:%s[.%s[.%s]]
2037	não há permissão para VACUUM do SQL.
2043	Tabela '%s': a função de indexação retornou um plano inválido.
2044	Máximo de %d tabelas em uma junção.
2046	Não é possível adicionar uma coluna PRIMARY KEY.
2047	Não é possível adicionar uma coluna UNIQUE.
2048	Não é possível adicionar uma coluna NOT NULL com o valor NULL padrão.
2049	Não é possível adicionar uma coluna com um padrão que não seja constante.
2050	Não é possível adicionar uma coluna a uma exibição.
2051	Não há permissão para ANALYZE no SQL.
2052	Nome inválido: '%s'
2053	Não há permissão para ATTACH do SQL.
2054	%s '%s' não pode fazer referência a objetos do banco de dados '%s'
2055	Proibido o acesso a '[%s.]%s.%s'
2056	não autorizado.
2058	nenhuma exibição deste tipo: '[%s.]%s'
2060	O nome da tabela temporária deve ser não qualificado.
2061	Tabela ‘%s’ já existente.
2062	Já existe um índice chamado: '%s'
2064	Nome da coluna duplicado: '%s'
2065	A tabela ‘%s’ tem mais de uma chave primária.
2066	Só há permissão para AUTOINCREMENT em INTEGER PRIMARY KEY.
2067	Nenhuma sequência de intercalação deste tipo: '%s'
2068	Não há permissão para parâmetros em exibições.
2069	A exibição ‘%s’ está definida de maneira circular.
2070	Não é possível descartar a tabela '%s'.
2071	Use DROP VIEW para excluir a exibição ‘%s’
2072	Use DROP TABLE para excluir a tabela ‘%s’
2073	A chave externa em ‘%s’ deve referenciar apenas uma coluna da tabela ‘%s’
2074	O número de colunas na chave externa não corresponde ao número de colunas na tabela referenciada.
2075	Coluna desconhecida ‘%s’ na definição da chave externa.
2076	Não é possível indexar a tabela ‘%s’
2077	Não é possível indexar exibições.
2080	Cláusulas ON CONFLICT em conflito especificadas.
2081	Nenhum índice deste tipo: '%s'.
2082	Não é possível ignorar índice associado à restrição UNIQUE ou PRIMARY KEY.
2083	Não há permissão para BEGIN no SQL.
2084	Não há permissão para COMMIT no SQL.
2085	Não há permissão para ROLLBACK no SQL.
2086	Não foi possível abrir um arquivo de banco de dados temporário para armazenar tabelas temporárias.
2087	Não foi possível identificar o objeto a ser reindexado.
2088	Não é possível modificar a tabela ‘%s’.

2089	Mão é possível modificar ‘%s’ por se tratar de uma exibição.
2090	O número da variável deve estar entre ?0 e ?%d.
2092	Uso inadequado do agregado ‘%s’ com suavização.
2093	Nome da coluna ambíguo: '[%s.[%s.]]%s'
2094	Nenhuma função deste tipo: '%s'
2095	Número errado de argumentos para a função ‘%s’.
2096	Subconsultas proibidas em restrições CHECK.
2097	Parâmetros proibidos em restrições CHECK.
2098	A árvore de expressões é muito grande (profundidade máxima %d).
2099	RAISE() talvez só possa ser usada em um programa disparador.
2100	A tabela '%s' tem %d colunas, mas foram fornecidos %d valores.
2101	O esquema do banco de dados está bloqueado: '%s'
2102	Instrução muito longa.
2103	Não foi possível excluir/modificar a sequência de intercalação por causa de instruções ativas.
2104	Muitos bancos de dados anexados - máximo %d
2105	Não é possível usar ATTACH no banco de dados dentro da transação.
2106	O banco de dados ‘%s’ já está em uso.
2108	Os bancos de dados anexados devem usar a mesma codificação de texto do banco de dados principal.
2200	Sem memória.
2201	Não foi possível abrir um banco de dados.
2202	Não é possível usar DETACH no banco de dados dentro da transação.
2203	Não é possível desanexar um banco de dados: '%s'
2204	O banco de dados ‘%s’ está bloqueado.
2205	Não foi possível obter um bloqueio de leitura no banco de dados.
2206	[column|columns] '%s'[,'%s'] não são [unique|is] não exclusivas.
2207	Esquema do banco de dados malformado.
2208	Não há suporte para o formato de arquivo.
2209	Token não reconhecido: '%s'
2300	Não foi possível converter o valor de texto em valor numérico.
2301	Não foi possível converter valor da string em data
2302	Não foi possível o converter valor em ponto flutuante em inteiro sem perda de dados.
2303	Não é possível reverter a transação – instruções SQL em andamento
2304	Não é possível confirmar a transação – instruções SQL em andamento.
2305	Tabela do banco de dados bloqueada: '%s'
2306	Tabela somente leitura.
2307	String ou blob muito grande.
2309	Não é possível abrir coluna indexada para gravação.
2400	Não é possível abrir valor de tipo %s
2401	Nenhuma rowid deste tipo: %s
2402	Nome do objeto reservado para uso interno: '%s'
2403	Não é possível alterar a exibição ‘%s’
2404	O valor padrão da coluna ‘%s’ não é constante.
2405	Não autorizado a usar a função ‘%s’
2406	Uso inadequado da função agregada ‘%s’
2407	Uso inadequado da agregação: '%s'
2408	Nenhum banco de dados deste tipo: '%s'
2409	A tabela ‘%s’ não tem uma coluna chamada ‘%s’
2501	nenhum módulo deste tipo: '%s'
2508	Nenhum ponto de recuperação deste tipo: '%s'
2510	Não é possível reverter – não há transações ativas.
2511	Não é possível confirmar – não há transações ativas.

Capítulo 68: Adobe Graphics Assembly Language (AGAL)

O Adobe Graphics Assembly Language (AGAL) é uma linguagem de sombreador para a definição dos programas de renderização de vértice e fragmento. Os programas AGAL deverão ser carregados no contexto de renderização no formato de código de bytes binário descrito nesse documento.

Formato de código de byte AGAL

O código de bytes AGAl deve usar o formato Endian.LITTLE_ENDIAN.

Cabeçalho do código de bytes

O código de bytes AGAL deve começar com um cabeçalho de 7 bytes:

```
A0 01000000 A1 00 -- for a vertex program
A0 01000000 A1 01 -- for a fragment program
```

Deslocamento (bytes)	Tamanho (bytes)	Nome	Descrição
0	1	magia	deve ser 0xa0
1	4	versão	deve ser 1
5	1	ID do tipo de sombreador	deve ser 0xa1
6	1	tipo de sombreador	0 para um programa de vértice; 1 para um programa de fragmento

Símbolos

O cabeçalho é imediatamente seguido por qualquer número de símbolos. Cada símbolo tem o tamanho de 192 bits (24 bytes) e sempre tem o formato:

```
[opcode][destino][origem1][origem2 ou classificador]
```

Nem todos os opcodes usam todos esses campos. Os campos não utilizados devem ser definidos como 0.

Códigos de operação

O campo [opcode] tem 32 bits de tamanho e pode assumir um destes valores:

Nome	Opcode	Operação	Descrição
mov	0x00	move	move os dados de origem1 para destino, em nível de componente
add	0x01	add	destino = origem1 + origem2, em nível de componente
abaixo	0x02	subtract	destino = origem1 - origem2, em nível de componente
mul	0x03	multiplicar	destino = origem1 * origem2, em nível de componente
div	0x04	dividir	destino = origem1 / origem2, em nível de componente

Nome	Opcode	Operação	Descrição
rcp	0x05	recíproco	destino = 1/origem1, em nível de componente
mín.	0x06	mínimo	destino = mínimo (origem1, origem2), em nível de componente
máx.	0x07	maximum	destino = máximo (origem1, origem2), em nível de componente
frc	0x08	fracionário	destino = origem1 - base (flutuante) (origem1), em nível de componente
sqt	0x09	raiz quadrada	destino = sqrt (origem1), em nível de componente
rsq	0x0a	raiz recíproca	destino = 1/sqrt (origem1), em nível de componente
pow	0x0b	potência	destino = pow (origem1, origem2), em nível de componente
log	0x0c	logaritmo	destino = log_2 (origem1), em nível de componente
exp	0x0d	exponencial	destino = 2^origem1, em nível de componente
nrm	0x0e	normalize	destino = normalização (origem1), em nível de componente (produz só um resultado de três componentes; o destino deve ser mascarado como .xyz ou menos)
sen	0x0f	seno	destino = sen (origem1), em nível de componente
cos	0x10	cosseno	destino = cos (origem1), em nível de componente
crs	0x11	produto vetorial	destino.x = origem1.y * origem2.z - origem1.z * origem2.y destino.y = origem1.z * origem2.x - origem1.x * origem2.z destino.z = origem1.x * origem2.y - origem1.y * origem2.x (produz só um resultado de 3 componentes; o destino deve ser mascarado como .xyz ou menos)
dp3	0x12	produto de ponto	destino = origem1.x*origem2.x + origem1.y*origem2.y + origem1.z*origem2.z
dp4	0x13	produto de ponto	destino = origem1.x*origem2.x + origem1.y*origem2.y + origem1.z*origem2.z + origem1.w*origem2.w
abs	0x14	absoluto	destino = abs(origem1), em nível de componente
neg	0x15	negar	destino = -origem1, em nível de componente
sat	0x16	saturar	destino = máximo (mínimo (origem1,1), 0), em nível de componente
m33	0x17	matriz de multiplicação 3x3	destino.x = (origem1.x * origem2[0].x) + (origem1.y * origem2[0].y) + (origem1.z * origem2[0].z) destino.y = (origem1.x * origem2[1].x) + (origem1.y * origem2[1].y) + (origem1.z * origem2[1].z) destino.z = (origem1.x * origem2[2].x) + (origem1.y * origem2[2].y) + (origem1.z * origem2[2].z) (produz só um resultado de 3 componentes; o destino deve ser mascarado como .xyz ou menos)

Nome	Opcode	Operação	Descrição
m44	0x18	matriz de multiplicação 4x4	destino.x = (origem1.x * origem2[0].x) + (origem1.y * origem2[0].y) + (origem1.z * origem2[0].z) + (origem1.w * origem2[0].w) destino.y = (origem1.x * origem2[1].x) + (origem1.y * origem2[1].y) + (origem1.z * origem2[1].z) + (origem1.w * origem2[1].w) destino.z = (origem1.x * origem2[2].x) + (origem1.y * origem2[2].y) + (origem1.z * origem2[2].z) + (origem1.w * origem2[2].w) destino.w = (origem1.x * origem2[3].x) + (origem1.y * origem2[3].y) + (origem1.z * origem2[3].z) + (origem1.w * origem2[3].w)
m34	0x19	matriz de multiplicação 3x4	destino.x = (origem1.x * origem2[0].x) + (origem1.y * origem2[0].y) + (origem1.z * origem2[0].z) + (origem1.w * origem2[0].w) destino.y = (origem1.x * origem2[1].x) + (origem1.y * origem2[1].y) + (origem1.z * origem2[1].z) + (origem1.w * origem2[1].w) destino.z = (origem1.x * origem2[2].x) + (origem1.y * origem2[2].y) + (origem1.z * origem2[2].z) + (origem1.w * origem2[2].w) (produz só um resultado de 3 componentes; o destino deve ser mascarado como .xyz ou menos)
kil	0x27	kil/descarte (somente sombreador de fragmento)	Se o componente de origem escalar única for menor do que zero, o fragmento será descartado e não será desenhado no buffer de quadros. (O registro de destino deve ser definido para todos os 0)
tex	0x28	amostra de textura (somente sombreador de fragmento)	o destino iguala a carga a partir da textura origem2 em coordenadas origem1. Neste caso, origem2 deve estar no formato de mostruário.
sge	0x29	definir se maior ou igual	destino = origem1> = origem2? 1 : 0, em nível de componente
slt	0x2a	definir se menor que	destino = origem1 <origem2? > 1 : 0, em nível de componente
seq	0x2c	definir se igual	destino = origem1 == origem2? 1 : 0, em nível de componente
sne	0x2d	definir se não igual	destino = origem1!= source2 ? 1 : 0, em nível de componente

Em AGAL2, os códigos de operação a seguir foram introduzidos:

Nome	Opcode	Operação	Descrição
ddx	0x1a	derivada parcial em X	Carregar a derivada parcial em X da origem1 no destino.
ddy	1x0b	derivada parcial em Y	Carregar a derivada parcial em Y da origem1 no destino.
ife	0x1c	se igual a	Ignorar se origem1 for igual a origem2.
ine	0x1d	se não for igual a	Ignorar se origem1 não for igual a origem2.
ifg	1x0e	se for maior que	Ignorar se origem1 for maior que ou igual a origem2.
ifl	1x0f	se for menor que	Ignorar se origem1 for menor que a origem2.
els	0x20	else	Bloco else
eif	0x21	Endif	Fechar bloco if ou else.

Formato do campo de destino

O campo [destino] tem 32 bits de tamanho:

```
31.............................0
----TTTT----MMMMNNNNNNNNNNNNNNNN
```

T = Tipo de registro (4 bits)

M = Máscara de gravação (4 bits)

N = Número de registro (16 bits)

- = indefinido, deve ser 0

Formato do campo de origem

O campo [source] tem 64 bits de tamanho:

```
63..............................................................0
D------------QQ----IIII----TTTTSSSSSSSSOOOOOOOONNNNNNNNNNNNNNNN
```

D = Direto=0/Indireto=1 para direto Q e I são ignorados, 1 bit

Q = Seleção do componente de registro de índice (2 bits)

I = Tipo de registro de índice (4 bits)

T = Tipo de registro (4 bits)

S = Swizzle (8 bits, 2 bits por componente)

O = Desvio indireto (8 bits)

N = Número de registro (16 bits)

- = indefinido, deve ser 0

Formato do campo do classificador

A segundo campo de origem do opcode tex deve estar no formato [classificador], cujo tamanho é de 64 bits:

```
63..............................................................0
FFFFMMMMWWWWSSSSDDDD--------TTTT--------BBBBBBBBNNNNNNNNNNNNNNNN
```

N = Número de registro do classificador (16 bits)

B = Desvio de detalhe de nível de textura (LOD), inteiro assinado, escala em 8. O valor de ponto flutuante é de b/8.0 (8 bits)

T = Tipo de registro deve ser 5, Classificador (4 bits)

F = Filtro (0=nearest,1=linear) (4 bits)

M = Mipmap (0=disable,1=nearest, 2=linear)

W = Agrupamento (0=clamp,1=repeat)

S = Bits de sinalizador especial (deve ser 0)

D = Dimensão (0=2D, 1=Cube)

Registros de Programa

O número de registros usados depende do perfil Context3D usado. O número de registros e seu uso são definidos na tabela a seguir:

Nome **AGAL2** **AGAL3** **Uso**	**Valor**	**AGAL**	
Número por programa de fragmento Número por programa de vértice Número por programa de fragmento Número por programa de vértice		Número por programa de fragmento	Número por programa de vértice
Suporte de perfil Context 3D Padrão Padrão estendido		Abaixo do padrão	
Versão SWF 25 28 e superior		Abaixo de 25	

Nome **AGAL2** **AGAL3** **Uso**	**Valor**	**AGAL**	
Atributo NA 8 NA 16 Entrada do sombreador de vértice; ler um buffer de vértice especificado usando Context3D.setVertexBufferAt().	0	NA	8
Constante 64 250 200 250 Entrada do sombreador; definir usando a família de funções Context3D.setProgramConstants().	1	28	128

Nome **AGAL2** **AGAL3** **Uso**	**Valor**	**AGAL**	
Temporário 26 26 26 26 Registro temporário para computação, programa externo não acessível.	2	8	8
Saída 1 1 1 1 Saída do sombreador: em um programa de vértice, a saída é a posição de espaço do clipe; em um programa de fragmento, a saída é uma cor.	3	1	1

Nome **AGAL2** **AGAL3** **Uso**	**Valor**	**AGAL**	
Variação 10 10 10 10 A transferência interpolou dados entre os sombreadores de vértice e de fragmento. Os registros variados do programa de vértice são aplicados como entrada no programa de fragmento. Os valores são interpolados conforme a distância dos vértices de triângulo.	4	8	8

Nome **AGAL2** **AGAL3** **Uso**	**Valor**	**AGAL**	
Mostruário 16 NA 16 NA Entrada do sombreador de fragmento; leitura de uma textura especificada usando Context3D.setTextureAt().	5	8	NA
Registro do fragmento 1 NA 1 NA É somente para escrita, usado para reescrever o valor z (ou valor de profundidade) escrito no vertex shader.	6	NA	NA
Tokens 1024 2048		200	

O AGAL Mini Assembler mais recente pode ser encontrado aqui.